Efficient Algorithms for Large Scale Network Problems

by

Dawei Huang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2020

Doctoral Committee:

Professor Seth Pettie, Chair
Assistant Professor Mahdi Cheraghchi
Associate Professor Barzan Mozafari
Assistant Professor Viswanath Nagarajan

Dawei Huang

hdawei@umich.edu

ORCID iD: 0000-0002-4496-2354

ACKNOWLEDGMENTS

First I would like to express my gratitude to my advisor Seth Pettie, who is the best advisor I can ever hope for. Throughout my 6 years as his students, he provided me numerous counselling in research, teaching and in learning to become professional. He provided me every opportunity he can for me to establish my research career and network. I cannot overstate the importance of his teachings in my career.

I would also like to thank Professor Barzan Mazafari and Professor Alex Pothen, who have exposed me to a more practical side of computer science and guided me in some very interesting and successful projects. They also taught me how to become a good collaborator.

I want to thank all my collaborators and coauthors, Shang-En Huang, Tsvi Kopelowitz, Dong Young Yoon, Zhijun Zhang, Yixiang Zhang, Shivaram Gopal for every bit of endeavors in our projects. I want to thank my office-mates Fang-yi Yu, Biaoshuai Tao, Yuqing Kong, Huck Bennett, Yi-Jun Chang, Hsin-Hao Su and Jimmy Zhu, with whom I have learnt things in research and life that I cannot learn on my own.

Finally, I want to thank my parents who have backed me in my path for almost 3 decades and provided me help and counsel when paths become difficult. I cannot imagine becoming the person I am now without their love and help.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

ABSTRACT

In recent years, the growing scale of data has renewed our understanding of what is an efficient algorithm and poses many essential challenges for the algorithm designers. This thesis aims to improve our understanding of many algorithmic problems in this context. These include problems in communication complexity, matching theory, and approximate query processing for database systems.

We first study the fundamental and well-known question of SetIntersection in communication complexity. We give a result that incorporates the error probability as an independent parameter into the classical trade-off between round complexity and communication complexity. We show that any r-round protocol that errs with error probability 2^{-E} requires $\Omega(Ek^{1/r})$ bits of communication. We also give several almost matching upper bounds.

In matching theory, we first study several generalizations of the ordinary matching problem, namely the f-matching and f-edge cover problem. We also consider the problem of computing a minimum weight perfect matching in a metric space with moderate expansion. We give almost linear time approximation algorithms for all these problems.

Finally, we study the sample-based join problem in approximate query processing. We present a result that improves our understanding of the effectiveness and limitations in using sampling to approximate join queries and provides a guideline for practitioners in building AQP systems from a theory perspective.

CHAPTER 1

Introduction

In computer science, efficiency is one of the core objectives faced by both theorists and practitioners. In his seminal papers, Edmonds[59, 60] first proposed the complexity class P and defined *polynomial time* as a metric for efficiency, i.e., any algorithms that run in time polynomial in its input size is considered efficient. This metric has become the rule of thumbs in determining which problems are tractable for more than three decades.

After the 90s, the notion of "efficiency" has quickly evolved due to the advancement of network technology and the growing scale of computer systems. The size of computer networks has multiplied from hundreds of nodes in the 80s, to millions of nodes in the 90s. By 2020, the internet contains more than 10^{10} nodes, and inputs to classical graph algorithms often consist of millions of nodes and billions of edges. For large scale inputs, complexities such as $O(n^3)$ or $O(n^2)$ are no longer considered efficient or even practical. For this reason, there has been a significant interest in designing extremely efficient algorithms that run in linear or almost linear time in recent years, e.g., [55, 96, 104, 125, 95, 116].

In some other computing regime such as distributed computing and database management, communication and space can become a more precious and limited resource compared to time. This puts forward new challenges for algorithm designers to make efficient or even optimal use of these resources and for theorists to understand the limitations of algorithms in these contexts.

In this thesis, we study three problems in three different regimes of theoretical computer science that are motivated by the quest for more efficient or optimal algorithms. The first part studies several fundamental problems in communication complexity. The second part studies almost linear-time algorithms for different variants of matching and edge cover problems. The third part is dedicated to database join problems and studies the effectiveness and limitations of the sampling-based approach.

1.1 Communication Complexity

Communication complexity was first proposed by Yao[142, 143] and proved to be an indispensable tool for both proving lower bounds in many different models of computation, and providing efficient protocols for effective communication between multiple network parties. The model consists of two parties Alice and Bob. Alice holds input $x \in X$, and Bob holds $y \in Y$. Their goal is to compute a function $f : X \times Y \mapsto Z$ at the point (x, y) while minimizing their *interaction*. To compute $f(x, y)$, the two parties engage in a communication protocol that proceeds in synchronous rounds. In each round, Alice and Bob take turns to transmit a message to the other party. Each message is allowed to depend on the sender's input as well as all previous messages. The receiver of the last message declares the output of the protocol.

We often use two metrics to measure the amount of interaction in a protocol. Given a protocol Π, the *communication complexity* is the worst-case total messages length of the protocol, while the *round complexity* is the number of messages exchanged of the protocol. Protocols can also be randomized, i.e., each message can also be dependent on a string of public or private random bits. In this case, we also use the *error probability*, i.e., the worst-case probability that the protocol does not output $f(x, y)$ given input (x, y) over all possible inputs, to measure the performance of the protocol.

In this thesis, we study two of the most well-known and well-understood problems in communication complexity: the SetIntersection problem and the EqualityTesting problem. In SetIntersection, the input are two sets $S, T \subset \mathcal{U}$ of size at most k in some universe $\mathcal{U}$ and the goal is to compute $S \cap T$. The problem finds many applications in computer science, especially in a distributed database, such as computing join, finding duplicates, and measuring Jaccard similarity[50]. Its lower bounds also find surprising applications in streaming algorithms[13] and combinatorial auctions[115].

A closely related problem is the EqualityTesting problem. Here Alice and Bob are given two k-dimensional vectors $x, y \in \mathcal{U}^k$ and they wish to decide for each $i \in [k]$, whether $x_i = y_i$. Notice that this problem can be seen as the *direct sum* of the Equality problem, where we have input $x, y \in \mathcal{U}$ and wish to decide whether $x = y$. Given a communication problem for computing a function f, the direct sum problem asks whether the communication complexity for computing k independent instance of f is k times the complexity for computing one single copy of f.

We also study two potentially easier versions of the problems, the SetDisjointness problem and the ExistsEqual problem. The SetDisjointness problem has the same

input as SetIntersection except that the goal is to determine whether S and T are disjoint or not. Similarly, the ExistsEqual problem is the corresponding problem for EqualityTesting that ask whether there exists an $i \in [k]$ such that $x_i = y_i$.

The deterministic complexity for these problems is well understood, see [103]. For randomized complexity, Kalyanasundaram and Schnitger[91] showed that the communication complexity for the SetDisjointness is at least $\Omega(k)$. Håstad and Wigderson [85] gave a matching $O(k)$-bit and $O(\log k)$-round protocol. In the seminal work of Sağlam and Tardos [130], as well as independently by Brody et al. [23], they gave a tight trade-off between communication complexity and round complexity for these problems, showing that any r-round protocols for ExistsEqual or SetDisjointness must use $O(k \log^{(r)} k)$ bits. Their lower bounds hold for protocols that succeed with $\Omega(1)$ probability, and the upper bound has failure probability at least $\exp(-\sqrt{k})$. This implies there is an optimal protocol with communication complexity $O(k)$ and round complexity $\log^* k$. It is easy to show that the error probability of such protocol is at least $\exp(-O(k))$ [103]. However, it is open that whether we can improve the error probability from $\exp(-\sqrt{k})$ to $\exp(-\Theta(k))$.

In Chapter 2, we address this problem, providing a tight three-way trade-off between communication complexity, round complexity, and error probability for ExistsEqual and SetDisjointness, and an almost tight trade-off for EqualityTesting and SetIntersection. In particular, we showed that any r-round protocol that succeeds with probability at least $1 - 2^{-E}$ requires communication complexity at least $\Omega(Ek^{1/r})$. As a corollary, we show that there is no universally optimal protocol simultaneously achieves the optimal trade-off between round complexity and communication complexity, as well as between error probability and communication complexity. In particular, there do not exist protocols with communication complexity $O(k+E)$, round complexity $\log^* k$, and error probability 2^{-E}. We also show that all these problems do *not* exhibit the direct sum properties in the regime of extremely small error probability.

1.2 Matching Theory

1.2.1 Generalized Matching Problems

In Chapter 3, we consider variants of matching problems in sequential settings. Given a graph $G = (V, E)$, possibly having a weight function $w : E \mapsto \mathbb{R}$ on edges, a matching is a set of vertex disjoint edges. The MAXIMUM WEIGHT/CARDINALITY

MATCHING (MWM/MCM) problem asks for a matching with the maximum total weight/size, while a MINIMUM WEIGHT PERFECT MATCHING (MWPM)) problem asks for a minimum weight matching that matches every single vertex. The MWM and MWPM problems are a special case of the *degree constrained subgraph* problem, where at each vertex, we look for a subset of edges $F \subset E$ that satisfy a certain degree constraint. Another common degree constraint problem is the MINIMUM WEIGHT/CARDINALITY EDGE COVER (MWEC/MCEC) problem, which asks for a minimum weight/size subset of edges such that each vertex is adjacent to *at least* of edge in the subset. It is known that MWEC and MWM are reducible to each other for graphs with nonnegative weight, see [132].

The best algorithm for MCM is by Micali and Vazirani [110, 137, 138] that runs in time $O(m\sqrt{n})$. For integral weight, Duan Pettie and Su gave an $O(m\sqrt{n}\log(nW))$ [56]. For many years, $O(m\sqrt{n})$ seems to be a barrier for exact matching algorithm. Approximation is one way to bypass the barrier. For MCM, the $O(m\sqrt{n})$ algorithm by Micali and Vazirani [110, 137, 138] is secretly a $(1-\epsilon)$-approximate MCM algorithm that runs in $O(m/\epsilon)$ time. For MWM, Duan and Pettie [55] gave an $(1-\epsilon)$-approximate algorithm in time $O(m/\epsilon \log(1/\epsilon))$.

A possible generalization of the matching/edge cover problems is called f-matching and f-edge cover. In this problem, f is a nonnegative integral function on vertices, which specifies the degree constraint. An f-matching/f-edge cover/f-factor is a subset of edges where each vertex v is matched to at most/at least/exactly $f(v)$ edges. The state of the art approach to solving the MAXIMUM WEIGHT f-MATCHING problem or the MINIMUM WEIGHT f-EDGE COVER problem is via a two-steps reduction with an intermediate problem called UNCAPACITATED b-MATCHING. Such a reduction blows up the size of the graph and is not approximation preserving. Gabow [71] gave a direct algorithm for f-matching and f-factor in time $O(f(V)(m + n\log n))$.

We give an almost linear time $(1 \pm \epsilon)$-approximate algorithm for the MAXIMUM WEIGHT f-MATCHING problem and the MINIMUM WEIGHT f-EDGE COVER problem. This generalizes the result of Duan and Pettie [55] and is the first $(1\pm\epsilon)$-approximate algorithm for generalized matching/edge cover problems that run in almost linear time.

1.2.2 Perfect Matching with Metric Constraint

In Chapter 4, we study the approximate minimum weight perfect matching problem with metric weights. Many high profile applications for the MWPMproblem are

not intended for graphs with general weights, but for distance metrics induced by certain graphs or geometries. For example, the famous Chritofides algorithm for metric traveling salesman problem requires us to first compute a minimum spanning tree in the metric space, then compute an MWPMamong the odd degree vertices in the MST. To solve the Chinese Postman Problem, we look for an MWPMin a distance metric induced by the odd degree vertices in a graph. The canonical way to solving this type of problem is to compute an APSP in a graph to obtain the distance function, and then run any MWPMalgorithm, e.g., [75, 78, 56]. This makes solving the MWPM problem the bottleneck for a near-linear time implementation of the Christofides algorithm [33].

The graph metric minimum weight perfect matching can be defined as follows: Given a graph $G = (V, E)$, weighted or unweighted, and an even subset $T \subseteq V$ of terminals, compute a perfect matching for vertices in T where the weight on any pair (u, v), $u, v \in T$ is defined as the distance between u and v on G. This problem is equivalent to the minimum weight T-join problem, in which we want to find a subgraph $F \subseteq E$ such that T is precisely the set of vertices with odd degrees in the subgraph (V, F).

The state of the art approach for a nearly linear time approximation is by using the Goemans-Williamson algorithm [78] and its linear time implementation [40]. The algorithm solved the more general *degree constraint forest* problem and only gives a 2-approximation. For Euclidean metrics, e.g., when the graph is induced by n points on a Euclidean plane, Agarwal and Varadarajan [136] gave a $(1 + \epsilon)$-approximation with time complexity $O(n\epsilon^{-3} \log^6 n)$.

Growth of a Metric Expansion property is an important concept in the algorithmic study of metric space [5, 102, 87, 94, 25, 20]. It captures the geometry of the underlying metric space by characterizing the relation between the volume and diameter. Many problems such as nearest neighbor [94] or traveling salesman [20] are hard for general metrics. This is because general metrics simply do not provide enough geometric structure for exploitation. Yet instances that arise from practice, such as database queries or geometric optimization, are often far from general, and expansion, growth, and dimensionality are used to characterize the geometric properties of these instances.

We showed that the Graph metric minimum weight perfect matching problem admits an almost linear time $(1 - \epsilon)$-approximation if the metric has slightly superlinear growth. In particular, if every ball in the metric space with radius d

has volume at least $d \log^2 d$, we can obtain a $(1-\epsilon)$-approximation solution in time $O(2^{O(1/\epsilon)} m \log n)$ time. If the ball has volume is $d^{1+\tau}$ for some $\tau > 0$, such as in a Euclidean grid of dimension $1+\tau$, the running time improves to $O(m \log n (1/\epsilon)^{1/\tau})$.

1.3 Join

Join is a common operation in relational algebra which takes in two or more relations, and a subset of common attributes, called *join attributes* on these relations. The output of the join operation consists of all combinations of tuples in these relations where the join attributes satisfy certain join conditions, such as equality, inequality, or more complex relations. A simple example is a natural binary join, where we take in two relations R and S and output a relation $R \bowtie S$, which consists of all tuples $(r \in R, s \in S)$ with equal value on their shared attribute(s). Joins are ever-present in analytical queries. However, they are considered an expensive query in terms of execution times and storage consumption, especially when joining large tables. For instance, the join of two relations can be as large as the Cartesian product of the two relations, which is costly to produce and store.

Approximate Query Processing (AQP) studies how to provide approximate answers with certain quality guarantees using a fraction of the resources that are required to compute an exact answer. In recent years, approximate query processing has regained major attention due to the growing size of data sets and the separation of computing and storage resources over networks, all of which make it much more difficult to perform exact data analytics with high speed and small space [111]. AQP provides a way to achieve high efficiency and interactivity in scenarios when a perfect answer is not required. Some of the high profile applications in AQP include relational data management [6, 7], sensor networks [99], visualization [45] and Adhoc analytic [10].

Sampling is one of the most commonly used techniques in AQP. The idea for sample-based AQP is to execute the original query on a small sample of the original relation in order to obtain a fast and approximate answer. Due to the cost of executing a plain join operation, sampling provides a promising alternative for approximating join queries. There are two different approaches for sample-based join approximation, the online approach, and the offline approach. The former refers to the methods of sampling directly from the join at query time. This method often involves directly computing the join or building complex indexes beforehand [107], which defeat the purpose of sampling. The offline sampling approach [6, 8, 11, 77] samples from each

individual relation offline, and execute the query on the samples of the relations. Although the offline approach leads to higher speedups [93], it is shown that sampling schemes without access to a large amount of statistical information to the other relation produce samples with poor quality [30]. More specifically, join of uniform and independent samples are no longer an independent sample of the join.

In Chapter 5, we study the effectiveness as well as limitations of sampling in approximating join queries. We formalize the problem as the JOIN SAMPLING PROBLEM, which captures the notion of sample quality under two popular metrics, the number of tuples retained from the join, and the accuracy of the query estimator generated from the sample. We formalize a hybrid sampling scheme called *Stratified Universe-Bernoulli Sampling* (SUBS), which allows for a smooth combination of stratified, universe, and Bernoulli sampling. We observed the connection between approximating join queries and the one-way communication complexity of approximate set intersection and gave an impossibility result for the join sampling problem. We also showed that under optimal parameters, the performance of the SUBS sampling is optimal with respect to our lower bound.

CHAPTER 2

Communication Complexity

2.1 Introduction

Communication Complexity was defined by Yao [143] in 1979 and has become an indispensible tool for proving lower bounds in models of computation in which the notions of *parties* and *communication* are not direct. See, e.g., books and monographs [129, 127, 103] and surveys [29, 108] on the subject. In this chapter we consider some of the most fundamental and well-studied problems in this model, such as SetDisjointness, SetIntersection, ExistsEqual, and EqualityTesting. Let us briefly define these problems formally since the terminology is not completely standard.

SetDisjointness and SetIntersection. In the SetDisjointness problem Alice and Bob receive sets $A \subset U$ and $B \subset U$ where $|A|, |B| \leq k$ and must determine whether $A \cap B = \emptyset$. Define $\mathsf{SetDisj}(k, r, p_{\text{err}})$ to be the minimum communication complexity of an r-round randomized protocol for this problem that errs with probability at most p_{err}. We can assume that $|U| = O(k^2/p_{\text{err}})$ without loss of generality.[1] The input to the SetIntersection problem is the same, except that the parties must *report the entire set* $A \cap B$. Define $\mathsf{SetInt}(k, r, p_{\text{err}})$ to be the minimum communication complexity of an r-round protocol for SetIntersection.

EqualityTesting and ExistsEqual. In the EqualityTesting problem Alice and Bob hold vectors $\mathbf{x} \in U^k$ and $\mathbf{y} \in U^k$ and must determine, for each index $i \in [k]$, whether $x_i = y_i$ or $x_i \neq y_i$. A potentially easier version of the problem, ExistsEqual, is to determine if there *exists* at least one index $i \in [k]$ for which $x_i = y_i$. Define $\mathsf{Eq}(k, r, p_{\text{err}})$ to be the randomized communication complexity

[1] Before the first round of communication, pick a pairwise independent $h : U \mapsto [O(k^2/p_{\text{err}})]$ and check whether $h(A) \cap h(B) = \emptyset$ with error probability $p_{\text{err}}/2$. Thus, having SetDisj depend additionally on $|U|$ is somewhat redundant, at least when $|U|$ is large.

of any r-round protocol for EqualityTesting that errs with probability p_{err}, and $\exists\mathsf{Eq}(k, r, p_{\text{err}})$ the corresponding complexity of ExistsEqual. Once again, we can assume that $|U| = O(k/p_{\text{err}})$ without loss of generality.

The deterministic communication complexity of these problems is well understood [103],[2] so we consider randomized complexity exclusively. Although these problems are well studied [85, 63, 23, 130, 91], most prior work has focused on the relationship between *round complexity* and *communication volume*, and has paid comparatively little attention to the role of p_{err}.

History. Håstad and Wigderson [85] gave an $O(\log k)$-round protocol for SetDisjointness in which Alice and Bob communicate $O(k)$ bits, which matched an $\Omega(k)$ lower bound of Kalyanasundaram and Schnitger [91]; see also [128, 24, 49]. Feder et al. [63] proved that EqualityTesting can be solved with $O(k)$ communication by an $O(\sqrt{k})$-round protocol that errs with probability $\exp(-\sqrt{k})$. Nikishkin [114] later improved the round complexity and the error probability to $\log k$ and $\exp(-k/\operatorname{polylog}(k))$. Improving [85], Sağlam and Tardos [130] gave an r-round protocol for SetDisjointness that uses $O(k \log^{(r)} k)$ communication, where $\log^{(r)}$ is the r-fold iterated logarithm function. For $r = \log^* k$ the error probability of this algorithm is $\exp(-\sqrt{k})$, coincidentally matching [63]. In independent work, Brody et al. [23] gave r-round and $O(r)$-round protocols for ExistsEqual and SetIntersection, respectively, that use $O(k \log^{(r)} k)$ communication and err with probability $1/\operatorname{poly}(k)$.

Sağlam and Tardos [130] were the first to show that this $O(k \log^{(r)} k)$ round vs communication tradeoff is optimal, using a combinatorial round elimination technique. In particular, this lower bound applies to any ExistsEqual protocol with constant error probability. Independently, Brody et al. [22, 23] gave a simpler proof for the EqualityTesting problem with the same tradeoff curve, but the lower bound assumes an error probability of $1/\operatorname{poly}(k)$. Brody et al. [23] also introduced a *randomized* reduction from SetIntersection to EqualityTesting, which carries a probability of error that is only tolerable if $p_{\text{err}} > \exp(-\tilde{O}(\sqrt{k}))$.

2.1.1 Contributions

First, we observe that a simple *deterministic* reduction shows that SetIntersection is essentially equivalent to EqualityTesting for any p_{err}, up to one round of communication,

[2] When $p_{\text{err}} = 0$, the deterministic complexity must be expressed in terms of k and $|U|$.

and SetDisjointness is essentially equivalent to ExistsEqual for any p_{err}. Theorem 2.1 is proved in Section 2.7; it is inspired by the randomized reduction of Brody et al. [23].

Theorem 2.1. *For any parameters $k \geq 1, r \geq 1$, and $p_{\text{err}} > 0$, it holds that*

$$\mathsf{Eq}(k, r, p_{\text{err}}) \leq \mathsf{SetInt}(k, r, p_{\text{err}}), \qquad \mathsf{SetInt}(k, r+1, p_{\text{err}}) \leq \mathsf{Eq}(k, r, p_{\text{err}}) + \zeta,$$
$$\exists\mathsf{Eq}(k, r, p_{\text{err}}) \leq \mathsf{SetDisj}(k, r, p_{\text{err}}), \qquad \mathsf{SetDisj}(k, r+1, p_{\text{err}}) \leq \exists\mathsf{Eq}(k, r, p_{\text{err}}) + \zeta,$$

where $\zeta = O(k + \log\log p_{\text{err}}^{-1})$.

Second, we prove that in any of the four problems, it is impossible to simultaneously achieve communication volume $O(k + \log p_{\text{err}}^{-1})$ in $O(\log^* k)$ rounds for all k, p_{err}. Specifically, if $p_{\text{err}} = 2^{-E}$, any r-round protocol needs $\Omega(Ek^{1/r})$ communication. In other words, if we insist on having $O(k)$ communication and $O(\log^* k)$ rounds, the smallest error probability that can be achieved is $p_{\text{err}} = \exp(-k^{1-\Theta(1/\log^* k)})$. We complement this lower bound with an upper bound showing that in $r + \log^*(k/E)$ rounds, we can solve EqualityTesting with $O(k + rEk^{1/r})$ communication. This matches our lower bound when $E \geq k$ and r is constant, but is slightly suboptimal when $r = \omega(1)$. We illustrate two ways to shave off this factor of r. We give an $(r + \log^*(k/E))$-round ExistsEqual protocol that communicates $O(k + Ek^{1/r})$ bits, as well as an EqualityTesting protocol that communicates $O(k + Ek^{1/r})$ bits, but with round complexity $O(r) + \log^*(k/E)$.

Our original interest in SetIntersection came from distributed subgraph detection in CONGEST[3] networks, which has garnered significant interest in recent years [26, 27, 89, 4, 54, 101, 64, 48, 79]. Izumi and LeGall [89] proved that triangle enumeration[4] requires $\Omega(n^{1/3}/\log n)$ rounds in the CONGEST model, and further showed that *local* triangle enumeration[5] requires $\Omega(\Delta/\log n)$ rounds in CONGEST, which can be as large as $\Omega(n/\log n)$.

The most natural way to solve (local) triangle enumeration is, for every edge $\{u, v\} \in E(G)$, to have u and v run a two-party SetIntersection protocol in which they compute $N(u) \cap N(v)$, where $N(u) = \{\mathsf{ID}(x) \mid \{u, x\} \in E(G)\}$ and $\mathsf{ID}(x) \in$

[3]In the CONGEST model there is a graph $G = (V, E)$ whose vertices are identified with processors and whose edges represent bidirectional communication links. Each vertex v does not know G, and is only initially aware of an $O(\log n)$-bit $\mathsf{ID}(v)$, $\deg(v)$, and global parameters $n \geq |V|$ and $\Delta \geq \max_{u \in V} \deg(u)$. Communication proceeds in synchronized rounds; in each round, each processor can send a (different) $O(\log n)$-bit message to each of its neighbors.

[4]Every triangle (3-cycle) in G must be reported by *some* vertex.

[5]Every triangle in G must be reported by at least one of the three constituent vertices. Izumi and LeGall [89] only stated the $\Omega(n/\log n)$ lower bound but it can also be expressed in terms of Δ.

$\{0,1\}^{O(\log n)}$ is x's unique identifier. Any r-round protocol with communication volume $O(\Delta)$ can be simulated in CONGEST in $O(\Delta/\log n + r)$ rounds since the message size is $O(\log n)$ bits. However, to guarantee a *global* probability of success at least $1 - 1/\operatorname{poly}(n)$, the failure probability of each SetIntersection instance must be $p_{\text{err}} = 2^{-E}$, $E = \Theta(\log n)$, which is independent of Δ. Our communication complexity lower bound suggests that to achieve this error probability, we would need $\Omega((\Delta + E\Delta^{1/r})/\log n + r)$ CONGEST rounds, i.e., with $r = \log \Delta$ we should not be able to do better than $O(\Delta/\log n + \log \Delta)$. We prove that (local) triangle enumeration can actually be solved *exponentially* faster, in $O(\Delta/\log n + \log\log \Delta)$ CONGEST rounds, without *necessarily* solving every SetIntersection instance.

Organization. The proof of Theorem 2.1 on the near-equivalence of SetIntersection/SetDisjointness and EqualityTesting/ExistsEqual appears in Section 2.7. Section 2.2 reviews concepts from information theory and communication complexity. In Section 2.3 we present new lower bounds for both EqualityTesting and ExistsEqual that incorporate rounds, communication, and error probability. Section 2.4 presents nearly matching upper bounds for EqualityTesting and ExistsEqual, and Section 2.6 applies them to the distributed triangle enumeration problem. We conclude with some open problems in Section 2.8.

2.2 Preliminaries

2.2.1 Notational Conventions

The set of positive integers at most t is denoted $[t]$. Random variables are typically written as capital letters (X, Y, M, etc.) and the values they take on are lower case (x, y, m, etc.). The letters $p, q, \mu, \mathcal{D}$ are reserved for probability mass functions (*p.m.f.*). E.g., $\mathcal{D}(x)$ denotes the probability that $X = x$ whenever $X \sim \mathcal{D}$. The *support* $\operatorname{supp}(\mathcal{D})$ of a distribution $\mathcal{D}$ is the set of all x for which $\mathcal{D}(x) > 0$. If $\mathcal{X} \subseteq \operatorname{supp}(\mathcal{D})$, $\mathcal{D}(\mathcal{X}) = \sum_{x \in \mathcal{X}} \mathcal{D}(x)$.

Many of our random variables are vectors. If x is a k-dimensional vector and $I \subseteq [k]$, x_I is the projection of x onto the coordinates in I and x_i is short for $x_{\{i\}}$. Similarly, if $\mathcal{D}$ is the p.m.f. of a k-dimensional random variable, $\mathcal{D}_I$ is the marginal distribution of $\mathcal{D}$ on the index set $I \subseteq [k]$.

Throughout the paper, log and exp are the base-2 logarithm and exponential

functions, and $\log^{(r)}$ and $\exp^{(r)}$ their r-fold iterated versions:

$$\log^{(0)}(x) = \exp^{(0)}(x) = x, \quad \log^{(r)}(x) = \log(\log^{(r-1)}(x)), \quad \exp^{(r)}(x) = \exp(\exp^{(r-1)}(x)).$$

The log-star function is defined to be $\log^*(x) = \min\{r \mid \log^{(r)}(x) \leq 1\}$. In particular, $\log^*(x) = 0$ if $x \leq 1$.

2.2.2 Information Theory

The most fundamental concept in information theory is *Shannon entropy*. The Shannon entropy of a discrete random variable X is defined as

$$\mathrm{H}(X) = -\sum_{x \in \mathrm{supp}(X)} \Pr[X = x] \log \Pr[X = x].$$

Since there may be cases in which different distributions are defined for the "same" random variable, we use $\mathrm{H}(p)$ in place of $\mathrm{H}(X)$ if X is drawn from a p.m.f. p. We also write $\mathrm{H}(\alpha)$, $\alpha \in (0, 1)$, to be the entropy of a Bernoulli random variable with success probability α. In general, we freely use a random variable and its p.m.f. interchangeably.

The *joint entropy* $\mathrm{H}(X, Y)$ of two random variables X and Y is simply

$$\mathrm{H}(X, Y) = -\sum_{x \in \mathrm{supp}(X)} \sum_{y \in \mathrm{supp}(Y)} \Pr[X = x \wedge Y = y] \log \Pr[X = x \wedge Y = y].$$

This notion can be easily extended to cases of more than two random variables. Here, we state a well known fact about joint entropy.

Fact 2.2. *For any random variables $X_1, X_2, \ldots, X_n$, their joint entropy is at most the sum of their individual entropies, i.e., $\mathrm{H}(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} \mathrm{H}(X_i)$.*

The *conditional entropy* of Y conditioned on another random variable X, denoted $\mathrm{H}(Y \mid X)$, measures the expected amount of extra information required to fully describe Y if X is known. It is defined to be

$$\begin{aligned}
\mathrm{H}(Y \mid X) &= \mathrm{H}(X, Y) - \mathrm{H}(X) \\
&= -\sum_{x \in \mathrm{supp}(X)} \Pr[X = x] \sum_{y \in \mathrm{supp}(Y)} \Pr[Y = y \mid X = x] \log \Pr[Y = y \mid X = x] \\
&\geq 0,
\end{aligned}$$

which can be viewed as a weighted sum of entropies of a number of conditional distributions.

Finally, the *mutual information* $\mathrm{I}(X\ ;\ Y)$ between two random variables X and Y quantifies the amount of information that is revealed about one random variable through knowing the other one:

$$\begin{aligned}
&\mathrm{I}(X\ ;\ Y)\\
&= \mathrm{H}(X) - \mathrm{H}(X \mid Y)\\
&= \mathrm{H}(X) + \sum_{y \in \mathrm{supp}(Y)} \Pr[Y = y] \sum_{x \in \mathrm{supp}(X)} \Pr[X = x \mid Y = y] \log \Pr[X = x \mid Y = y].
\end{aligned}$$

2.2.3 Communication Complexity

Let $f(x, y)$ be a function over domain $\mathcal{X} \times \mathcal{Y}$, and consider any two-party communication protocol $Q(x, y)$ that computes $f(x, y)$, where one party holds x and the other holds y. The *transcript* of Q on (x, y) is defined to be the concatenation of all messages exchanged by the two parties, in order, as they execute on input (x, y). The *communication cost* of Q is the maximum transcript length produced by Q over all possible inputs.

Let Q_d be a deterministic protocol for f and suppose μ is a distribution over $\mathcal{X} \times \mathcal{Y}$. The *distributional error probability of Q_d with respect to μ* is the probability $\Pr_{(x,y)\sim\mu}[Q_d(x, y) \neq f(x, y)]$. For any $0 < \epsilon < 1$, the (μ, ϵ)-*distributional deterministic communication complexity* of the function f is the minimum communication cost of any protocol Q_d that has distributional error probability at most ϵ with respect to the distribution μ.

A randomized protocol $Q_r(x, y, w)$ also takes a public random string $w \sim \mathcal{W}$ as input. The error probability of Q_r is calculated as $\max_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \Pr_{w\sim\mathcal{W}}[Q_r(x, y, w) \neq f(x, y)]$. The ϵ-*randomized communication complexity* of f is the minimum communication cost of Q_r over all protocols Q_r with error probability at most ϵ.

Yao's *minimax principle* [142] is a common starting point for lower bound proofs in randomized communication complexity. The easy direction of Yao's minimax principle states that the communication cost of the best deterministic protocol specific to any particular distribution is at most the communication cost of any randomized protocol on its worst case input.

Lemma 2.3 (Yao's minimax principle [142])**.** *Let $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{Z}$ be the function to be computed. Let $D_{\mu,\epsilon}(f)$ be the (μ, ϵ)-distributional deterministic communication*

complexity of f, and let $R_\epsilon(f)$ be the ϵ-randomized communication complexity of f. Then for any $0 < \epsilon < 1/2$,

$$\max_\mu D_{\mu,\epsilon}(f) \leq R_\epsilon(f).$$

Therefore, to show a lower bound on the ϵ-randomized communication complexity of a function f, it suffices to find a hard distribution μ on the input set and prove a lower bound for the communication cost of any deterministic protocol that has distributional error probability at most 2ϵ with respect to μ.

2.3 Lower Bounds on ExistsEqual and EqualityTesting

In this section we prove lower bounds on EqualityTesting and ExistsEqual. Theorem 2.4 obviously follows directly from Theorem 2.5, but we prove them in that order nonetheless because Theorem 2.4 is a bit simpler.

Theorem 2.4. *Any r-round randomized protocol for EqualityTesting on vectors of length k that errs with probability $p_{\text{err}} = 2^{-E}$ requires at least $\Omega(Ek^{1/r})$ bits of communication.*

Theorem 2.5. *Any r-round randomized protocol for ExistsEqual on vectors of length k that errs with probability $p_{\text{err}} = 2^{-E}$ requires at least $\Omega(Ek^{1/r})$ bits of communication.*

Without any constraint on the number of rounds, EqualityTesting trivially requires $\Omega(k)$ communication. ExistsEqual also requires $\Omega(k)$ communication, through a small modification to the SetDisjointness lower bounds [91, 128]. Even when $k = 1$, we need at least $\Omega(E)$ communication to solve EqualityTesting/ExistsEqual with error probability 2^{-E} [103]. Thus, we can assume that $E = \Omega(k^{1-1/r})$, $k^{1/r} = \Omega(1)$, and hence $r = O(\log k)$. For example, some calculations later in our proof hold when $r \leq (\log k)/6$. When proving Theorem 2.5, we will further assume $E = \Omega(\log k)$ when $r = 1$, which is reasonable because of Sağlam and Tardos' $\Omega(k \log^{(r)} k) = \Omega(k \log k)$ lower bound [130].

2.3.1 Structure of the Proof

We consider deterministic strategies for ExistsEqual/EqualityTesting when Alice and Bob pick their input vectors independently from the uniform distribution on $[t]^k$, where $t = 2^{cE}$ and $c = 1/2$. Although the probability of seeing a collision in any particular coordinate is small, it is still much larger than the tolerable error probability

(since $c < 1$), so it is incorrect to declare "not equal in every coordinate" without performing any communication.

We suppose, for the purpose of obtaining a contradiction, that there is a protocol for EqualityTesting with error probability 2^{-E} and communication complexity $c'Ek^{1/r}$, where $c' = c/100$. The length of the jth message is l_j, which could depend on the parameters (E, r, k, etc.) and possibly in some complicated way on the transcript of the protocol before round j.[6]

Our proof must necessarily consider transcripts of the protocol that are extremely unlikely (occurring with probability close to 2^{-E}) and also maintain a high level of uncertainty about *which* coordinates of Alice's and Bob's vectors might be equal. Consider the first message. Alice picks her input vector $x \in [t]^k$, which dictates the first message m_1. Suppose, for simplicity, that it betrays exactly $l_1/k < c'Ek^{1/r-1}$ bits of information per coordinate of x. Before Bob can respond with a message m_2 he must commit to his input, say y. Most values of y result in "good" outcomes: nearly all non-equal coordinates get detected immediately and the effective size of the problem is dramatically reduced. We are not interested in these values of y, only very "bad" values. Let I_1 be the first $k^{1-1/r}$ coordinates (or, more generally, $k^{1-1/r}$ coordinates that m_1 revealed below-average information about). With probability about $(2^{-c'Ek^{1/r-1}})^{|I_1|} = 2^{-c'E}$, Bob picks an input y that is *completely consistent* with Alice's on I_1, i.e., as far as he can tell $y_i = x_i$ for every $i \in I_1$. Rather than sample y uniformly from $[t]^k$, we sample it from a "hybrid" distribution: y_{I_1} is sampled from the same distribution that m_1 revealed about x_{I_1} (forcing the above event to happen with probability 1), and $y_{[k]\setminus I_1}$ is sampled from Bob's former distribution (in this case, the uniform distribution on $[t]^{k-|I_1|}$), conditioned on the value of y_{I_1}.

This process continues round by round. Bob's message m_2 betrays at most $l_2/|I_1| < c'Ek^{2/r-1}$ bits of information on each coordinate of y_{I_1}, and there must be an index set $I_2 \subset I_1$ with $|I_2| = k^{1-2/r}$ such that, with probability around $2^{-c'E}$, it is completely consistent that $x_{I_2} = y_{I_2}$. Alice resamples her input so that this (rare) event occurs with probability 1, generates m_3, and continues.

At the end of this process $|I_r| = k^{1-r/r} = 1$, and yet Alice and Bob have revealed less than the full cE bits of entropy about x_{I_r} and y_{I_r}. Regardless of whether they report "equal" or "not equal" (on I_r), they are wrong with probability greater than 2^{-E}. Are we done? Absolutely not! The problem is that this strange process for

[6] In the context of ExistsEqual/EqualityTesting, it is natural to think about uniform-length messages, $l_j = c'Ek^{1/r}/r$, or lengths that decay according to some convergent series, e.g., $l_j \propto c'Ek^{1/r}/2^j$ or $l_j \propto c'Ek^{1/r}/j^2$.

sampling a possible transcript of the protocol might itself only find transcripts that occur with probability $\ll 2^{-E}$, making any conclusions we make about its (probability of) correctness moot. Generally speaking, we need to show that Alice's and Bob's actions are consistent with events that occur with probability $\gg 2^{-E}$.

Let us first make every step of the above process a bit more formal. It is helpful to think about Alice's and Bob's inputs not being *fixed* vectors selected at time zero, but simply distributions over vectors that change as messages progressively reveal more information about them.

- Before the jth round of communication, the sender of the jth message's input is drawn from a discrete distribution $\widehat{\mathcal{D}}^{(j-1)}$ over $[t]^k$. The receiver of the jth message's input is drawn from the distribution $\mathcal{D}^{(j-1)}$. For example, when $j = 1$, if Alice speaks first then her initial distribution, $\widehat{\mathcal{D}}^{(0)}$, and Bob's initial distribution, $\mathcal{D}^{(0)}$, are both uniform over $[t]^k$.

- Before the jth round of communication both parties are aware of an index set I_{j-1} such that, informally, (i) the distributions $\mathcal{D}^{(j-1)}_{I_{j-1}}$ and $\widehat{\mathcal{D}}^{(j-1)}_{I_{j-1}}$ are very similar, and in particular, it is consistent that their inputs are identical on I_{j-1}, and (ii) the messages transmitted so far reveal "average" or below-average information about these coordinates. For example, $I_0 = [k]$ and it is consistent with the empty transcript that Alice's and Bob's inputs are identical on every coordinate.

- The jth message is a random variable $M_j \in \{0,1\}^{l_j}$. In order to pick an m_j according to the right distribution, the sender picks an input $x \sim \widehat{\mathcal{D}}^{(j-1)}$ which, together with the history $m_1, \ldots, m_{j-1}$, determines m_j. The sender transmits m_j to the receiver and promptly forgets x. The sender's new distribution (i.e., $\widehat{\mathcal{D}}^{(j-1)}$, conditioned on $M_j = m_j$) is called $\mathcal{D}^{(j)}$.

- The distribution $\mathcal{D}^{(j)}$ may reveal information about the coordinates I_{j-1} in an irregular fashion. We find a subset $I_j \subset I_{j-1}$ of coordinates, $|I_j| = k^{1-j/r}$, for which the amount of information revealed by $\mathcal{D}^{(j)}_{I_j}$ is at most average. The receiver of m_j changes his input distribution to $\widehat{\mathcal{D}}^{(j)}$, which is defined so that it basically agrees with $\mathcal{D}^{(j)}_{I_j}$ and the marginal distribution $\widehat{\mathcal{D}}^{(j)}_{[k]\setminus I_j}$, conditioned on the value selected by $\mathcal{D}^{(j)}_{I_j}$, is identical to $\mathcal{D}^{(j-1)}_{[k]\setminus I_j}$.

- The reason $\mathcal{D}^{(j)}_{I_j}$ and $\widehat{\mathcal{D}}^{(j)}_{I_j}$ are not *identical* is due to two filtering steps. To generate $\widehat{\mathcal{D}}^{(j)}$, we remove points from the support that have tiny (but non-zero) probability, which may be too close to the error probability. Intuitively

these rare events necessarily represent a small fraction of the probability mass. Second, we remove points from the support if the ratio of their probability occurring under $\mathcal{D}^{(j)}$ over $\mathcal{D}^{(j-1)}$ is too high. Intuitively, we want to conclude that if there is a high probability of an error occurring under $\mathcal{D}^{(j)}$ then the probability is also high under $\mathcal{D}^{(j-1)}$ (and by unrolling this further, under $\mathcal{D}^{(0)}$). This argument only works if the ratios are what we would expect, given how much information is being revealed about these coordinates by m_j. As a result of these two filtering steps, $\mathcal{D}^{(j)}_{I_j}(x_{I_j})$ and $\widehat{\mathcal{D}}^{(j)}_{I_j}(x_{I_j})$ differ by at most a constant factor, for any particular vector $x_{I_j} \in [t]^{|I_j|}$.

2.3.2 A Lower Bound on EqualityTesting

We begin with two general lemmas about discrete probability distributions that play an important role in our proof.

Roughly speaking, Lemma 2.6 captures and generalizes the following intuition: Suppose p is a *high entropy* distribution on some universe U and q is obtained from p by conditioning on an event $\mathcal{X} \subseteq U$ such that $p(\mathcal{X})$ is large, say some constant like $1/4$. If p's entropy is close to $\log |U|$, then q's entropy should not be much smaller than that of p. As our proof goes on round by round, we will constantly throw away part of the input distribution's support to meet certain conditions. It is Lemma 2.6 that guarantees that the input distributions continue to have relatively high entropy.

Lemma 2.7 comes into play because the error probability will be calculated backward in a round-by-round manner. Suppose the old distribution (p) has no extremely low probability point and the new distribution (q) has almost full entropy. Lemma 2.7 provides us with a useful tool to transfer a lower bound on the probability of any event w.r.t. q to a lower bound on the same event w.r.t. p.

Lemma 2.6. *Let p and q be distributions defined on a universe of size 2^s. Suppose both of the following properties are satisfied:*

1. *The entropy of p is $\mathrm{H}(p) \geq s - g$, where $0 \leq g \leq s$;*
2. *There exists $0 < \alpha < 1$ such that $q(x) \leq p(x)/\alpha$ holds for every value $x \in \mathrm{supp}(q)$.*

The entropy of q is lower bounded by:

$$\mathrm{H}(q) \geq s - g/\alpha - \mathrm{H}(\alpha)/\alpha.$$

Proof. Let $\mathcal{X}$ be the whole universe. From our assumptions, the entropy of q can be lower bounded as follows.

$$\begin{aligned}
\mathrm{H}(q) &= \sum_{x\in\mathcal{X}} q(x)\log\frac{1}{q(x)} && \text{(Defn. of H}(q)\text{.)}\\
&= \frac{1}{\alpha}\sum_{x\in\mathcal{X}} \alpha q(x)\log\frac{1}{\alpha q(x)} + \log\alpha && (\textstyle\sum_{x\in\mathcal{X}} q(x) = 1.)\\
&\geq \frac{1}{\alpha}\sum_{x\in\mathcal{X}}\left[p(x)\log\frac{1}{p(x)} - (p(x)-\alpha q(x))\log\frac{1}{p(x)-\alpha q(x)}\right] + \log\alpha
\end{aligned}$$

The previous step follows from Assumption 2 and the fact that $x\log x^{-1} + y\log y^{-1} \geq (x+y)\log(x+y)^{-1}$ for any $x, y \geq 0$. Continuing,

$$\begin{aligned}
&\geq \frac{1}{\alpha}\left[s - g - \sum_{x\in\mathcal{X}}(p(x)-\alpha q(x))\log\frac{1}{p(x)-\alpha q(x)}\right] + \log\alpha && \text{(Assumption 1.)}\\
&\geq \frac{1}{\alpha}\left[s - g - (1-\alpha)\log\frac{2^s}{1-\alpha}\right] + \log\alpha && \text{(Concavity of logarithm.)}\\
&= s - \frac{g}{\alpha} + \frac{1-\alpha}{\alpha}\log(1-\alpha) + \log\alpha \;=\; s - \frac{g}{\alpha} - \frac{\mathrm{H}(\alpha)}{\alpha}.
\end{aligned}$$

□

Lemma 2.7. *Let p and q be distributions defined on a universe of size 2^s. Suppose both of the following properties are satisfied:*

1. *The entropy of q is $\mathrm{H}(q) \geq s - g_1$, where $0 \leq g_1 \leq s$;*
2. *There exists $g_2 \geq 0$ such that $p(x) \geq 2^{-s-g_2}$ holds for every value $x \in \mathrm{supp}(q)$.*

Then, for any $0 < \alpha < 1$,

$$\Pr_{x\sim q}\left[\frac{q(x)}{p(x)} > 2^{g_1/\alpha + g_2 - (1-\alpha)\log(1-\alpha)/\alpha}\right] \leq \alpha.$$

Proof. Let $\mathcal{X}_0 = \{x \in \mathrm{supp}(q) \mid q(x)/p(x) \leq 2^{g_1/\alpha+g_2-(1-\alpha)\log(1-\alpha)/\alpha}\}$ and $\mathcal{X}_1 = \mathrm{supp}(q)\setminus\mathcal{X}_0$. Suppose, for the purpose of obtaining a contradiction, that the conclusion of the lemma is false, i.e., $q(\mathcal{X}_1) = \alpha_0$, for some $\alpha_0 > \alpha$. Notice that for each value $x \in \mathcal{X}_1$, Assumption 2 implies that

$$q(x) > p(x)\cdot 2^{g_1/\alpha+g_2-(1-\alpha)\log(1-\alpha)/\alpha} \geq 2^{-s+g_1/\alpha-(1-\alpha)\log(1-\alpha)/\alpha}. \tag{2.1}$$

Then we can upper bound the entropy of q as follows.

$$\begin{aligned}
\mathrm{H}(q) &= \sum_{x\in\mathcal{X}_0} q(x)\log\frac{1}{q(x)} + \sum_{x\in\mathcal{X}_1} q(x)\log\frac{1}{q(x)} && \text{(Defn. of H}(q)\text{.)}\\
&< \sum_{x\in\mathcal{X}_0} q(x)\log\frac{1}{q(x)} + \alpha_0\left[s - \frac{g_1}{\alpha} + \frac{1-\alpha}{\alpha}\log(1-\alpha)\right] && \text{(Eqn. (2.1).)}\\
&\le (1-\alpha_0)\log\frac{2^s}{1-\alpha_0} + \alpha_0\left[s - \frac{g_1}{\alpha} + \frac{1-\alpha}{\alpha}\log(1-\alpha)\right]\\
& && \text{(Concavity of logarithm.)}\\
&= s - \frac{\alpha_0}{\alpha}\cdot g_1 + \alpha_0\left[\frac{1-\alpha}{\alpha}\log(1-\alpha) - \frac{1-\alpha_0}{\alpha_0}\log(1-\alpha_0)\right]\\
&< s - g_1,
\end{aligned}$$

where the last step follows from the monotonicity of $(1-\alpha)\log(1-\alpha)/\alpha$. This contradicts Assumption 1. □

We are now ready to begin the proof of Theorem 2.4 proper. Fix a round j and a particular history $(m_1,\ldots,m_{j-1})$ up to round $j-1$. We let $\mu_j(m_j)$ denote the probability that the jth message is m_j, if the input to the sender is drawn from $\widehat{\mathcal{D}}^{(j-1)}$. Define $\mathcal{D}^{(j)}[m_j]$ to be the new input distribution of the sender after he commits to m_j. When m_j is clear from context, it is denoted $\mathcal{D}^{(j)}$. (The process for deriving $\widehat{\mathcal{D}}^{(j)}$ from $\mathcal{D}^{(j)}$ and $\mathcal{D}^{(j-1)}$ on the receiver's end will be explained in detail later.)

We will prove by induction that the following Invariant 2.8 holds for each $j\in[0,r]$, where the particular values of I_j, $\mathcal{D}^{(j)}$, $\widehat{\mathcal{D}}^{(j)}$, and $l_1,\ldots,l_j$ depend on the transcript $m_1,\ldots,m_j$ that is sampled. In the base case, Invariant 2.8 clearly holds when $j=0, I_0=[k]$, and both $\widehat{\mathcal{D}}^{(0)}, \mathcal{D}^{(0)}$ are the uniform distribution over $[t]^k$.

Invariant 2.8. *After round $j\in[0,r]$ the partial transcript is $m_1,\ldots,m_j$, which determines the values $\{l_{j'}, \widehat{\mathcal{D}}^{(j')}, \mathcal{D}^{(j')}, I_{j'}\}_{j'\le j}$. The index set $I_j\subseteq[k]$ satisfies all of the following:*

1. $|I_j| = k^{1-j/r}$.
2. *Each value $x_{I_j}\in[t]^{|I_j|}$ satisfies $\widehat{\mathcal{D}}^{(j)}_{I_j}(x_{I_j}) \le 4\mathcal{D}^{(j)}_{I_j}(x_{I_j})$.*
3. *Each nonempty subset $I'\subseteq I_j$ satisfies*

$$\mathrm{H}(\widehat{\mathcal{D}}^{(j)}_{I'}) \ge \left(cE - \sum_{u=1}^{j}\frac{16^{j-u+1}l_u}{k^{1-(u-1)/r}} - 22^j\right)|I'|.$$

In accordance with our informal discussion in Section 2.3.1, I_j is a subset of indices on which both parties have learned little information about each other from the partial transcript $m_1, \ldots, m_j$. Invariant 2.8(2) ensures that the two parties draw their inputs after the jth round from similar distributions. Invariant 2.8(3) is the most important property. It says that the information revealed by $\widehat{\mathcal{D}}^{(j)}$ about I' is roughly what one would expect, given the message lengths $l_1, \ldots, l_j$. Note that the uth message conveys information about $|I_{u-1}| = k^{1-(u-1)/r}$ indices so the average information-per-index should be $l_u/k^{1-(u-1)/r}$. The factor 16^{j-u+1} and the extra term 22^j come from Lemma 2.6, which throws away part of the input distribution in each round, progressively distorting the distributions in minor ways.

To begin our induction, at round j we find a large fraction of possible messages m_j that reveal little information about the sender's input, projected onto I_{j-1}. This is possible because the length of the message $l_j = |m_j|$ reflects an upper bound on the expected information gain. This idea is formalized in the following Lemma 2.9.

Lemma 2.9. *Fix $j \in [1, r]$ and suppose Invariant 2.8 holds for $j-1$. Then there exists a subset of messages $\mathcal{M}'_j$ with $\mu_j(\mathcal{M}'_j) \geq 1/2$ such that each message $m_j \in \mathcal{M}'_j$ satisfies*

$$\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}[m_j]) \geq \left(cE - 2\sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{1-(u-1)/r}} - 2 \cdot 22^{j-1} \right) |I_{j-1}|.$$

Proof. Let $\mathcal{M}'_j$ contain all messages m_j satisfying the above inequality and $\overline{\mathcal{M}'_j}$ be its complement. Suppose, for the purpose of obtaining a contradiction, that the conclusion of the lemma is not true, i.e., $\mu_j(\overline{\mathcal{M}'_j}) = \alpha > 1/2$. Then the entropy of

$\widehat{\mathcal{D}}^{(j-1)}_{I_{j-1}}$ can be upper bounded as follows.

$$
\begin{aligned}
&\mathrm{H}(\widehat{\mathcal{D}}^{(j-1)}_{I_{j-1}}) \\
&= \mathrm{I}(\widehat{\mathcal{D}}^{(j-1)}_{I_{j-1}} \ ; \ M_j) + \sum_{m_j \in (\mathcal{M}'_j \cup \overline{\mathcal{M}'_j})} \mu_j(m_j)\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}[m_j]) && \text{(Defn. of } I(\cdot,\cdot).) \\
&\leq \mathrm{H}(M_j) + \sum_{m_j \in (\mathcal{M}'_j \cup \overline{\mathcal{M}'_j})} \mu_j(m_j)\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}[m_j]) && (\mathrm{I}(X \ ; \ \cdot) \leq \mathrm{H}(X).) \\
&\leq l_j + \sum_{m_j \in \mathcal{M}'_j} \mu_j(m_j)\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}[m_j]) + \sum_{m_j \in \overline{\mathcal{M}'_j}} \mu_j(m_j)\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}[m_j]) \\
& && (H(M_j) \leq |M_j| = l_j.) \\
&< l_j + (1-\alpha)cE|I_{j-1}| + \alpha\left(cE - 2\sum_{u=1}^{j} \frac{16^{j-u}l_u}{k^{1-(u-1)/r}} - 2 \cdot 22^{j-1}\right)|I_{j-1}| \\
& && \text{(Defn. of } \overline{\mathcal{M}'_j}.) \\
&= l_j + \left(cE - 2\alpha\sum_{u=1}^{j} \frac{16^{j-u}l_u}{k^{1-(u-1)/r}} - 2\alpha \cdot 22^{j-1}\right)|I_{j-1}| \\
&< \left(cE - \sum_{u=1}^{j-1} \frac{16^{j-u}l_u}{k^{1-(u-1)/r}} - 22^{j-1}\right)|I_{j-1}|, && \text{(Because } \alpha > 1/2.)
\end{aligned}
$$

This contradicts Invariant 2.8(3) at index $j-1$. □

After the jth message m_j is sent, the next step is to identify a set of coordinates I_j such that $\mathcal{D}^{(j)}$ still reveals little information about I_j *and every subset of* I_j, since we need this property to hold for $I_{j+1}, \ldots, I_r$ in the future, all of which are subsets of I_j. We also want I_j not to contain many low probability points w.r.t. $\mathcal{D}^{(j-1)}$, since this may stop us from applying Lemma 2.7 later on. These two constraints are captured by parts (2) and (1), respectively, of Lemma 2.10.

Lemma 2.10. *Fix $j \in [1,r]$ and suppose Invariant 2.8 holds for $j-1$. Then there exists a subset of messages $\mathcal{M}_j \subseteq \mathcal{M}'_j$ (from Lemma 2.9) with $\mu_j(\mathcal{M}_j) \geq 1/4$ such that for each message $m_j \in \mathcal{M}_j$, there exists a subset $I_j \subseteq I_{j-1}$ of size $|I_j| = k^{1-j/r}$ satisfying both of the following properties:*

1. $\Pr_{x_{I_j} \sim \mathcal{D}^{(j)}_{I_j}}\left[\mathcal{D}^{(j-1)}_{I_j}(x_{I_j}) < (4t)^{-|I_j|}/32\right] \leq 1/2$;
2. *Each nonempty subset $I' \subseteq I_j$ satisfies*

$$\mathrm{H}(\mathcal{D}^{(j)}_{I'}) \geq \left(cE - 4\sum_{u=1}^{j} \frac{16^{j-u}l_u}{k^{1-(u-1)/r}} - 4 \cdot 22^{j-1}\right)|I'|.$$

Proof. We first prove that for each message $m_j \in \mathcal{M}'_j$ (from Lemma 2.9), there exists a subset $J_0 \subseteq I_{j-1}$ of size $|J_0| \geq |I_{j-1}|/2$ such that each nonempty subset $I' \subseteq J_0$ satisfies part (2) of the lemma. Suppose $J_1, J_2, \ldots, J_w$ are disjoint subsets of I_{j-1}, each of which *violates* the inequality of part (2), whereas none of the subsets of $J_0 = I_{j-1} \setminus (\bigcup_{v=1}^{w} J_v)$ do. Then we can upper bound the entropy of $\mathcal{D}^{(j)}_{I_{j-1}}$ as follows.

$$\begin{aligned}
\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}) &\leq \sum_{v=0}^{w} \mathrm{H}(\mathcal{D}^{(j)}_{J_v}) && \text{(Fact 2.2.)} \\
&< cE|J_0| + \sum_{v=1}^{w} \left(cE - 4 \sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{1-(u-1)/r}} - 4 \cdot 22^{j-1} \right) |J_v| && \text{(Defn. of } J_v\text{.)} \\
&= cE|I_{j-1}| - 4|I_{j-1} \setminus J_0| \left(\sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{1-(u-1)/r}} + 22^{j-1} \right).
\end{aligned}$$

On the other hand, from Lemma 2.9, having $m_j \in \mathcal{M}'_j$ guarantees that

$$\mathrm{H}(\mathcal{D}^{(j)}_{I_{j-1}}) \geq \left(cE - 2 \sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{1-(u-1)/r}} - 2 \cdot 22^{j-1} \right) |I_{j-1}|.$$

The two inequalities above are only consistent if $|I_{j-1} \setminus J_0| \leq |I_{j-1}|/2$, or equivalently $|J_0| \geq |I_{j-1}|/2$. Thus, J_0 exists with the right cardinality, as claimed.

Now suppose, for the purpose of obtaining a contradiction, that the lemma is false. For every $m_j \in \mathcal{M}'_j$ there is a corresponding index set J_0 whose subsets satisfy part (2) of the lemma. If the lemma is false, that means there is a subset $\mathcal{M}''_j \subseteq \mathcal{M}'_j$ of "bad" messages with $\mu_j(\mathcal{M}''_j) > 1/4$ such that, for each $m_j \in \mathcal{M}''_j$, none of the $\binom{|J_0|}{|I_j|}$ choices for $I_j \subseteq J_0$ satisfy part (1) of the lemma. (Remember that J_0 depends on m_j but the lower bound on $|J_0| \geq |I_{j-1}|/2$ is independent of m_j.) Consider the following summation:

$$Z = \sum_{\substack{I_j \subseteq I_{j-1} : \\ |I_j| = k^{1-j/r}}} \sum_{\substack{x_{I_j} \in [t]^{|I_j|} : \\ \mathcal{D}^{(j-1)}_{I_j}(x_{I_j}) < (4t)^{-|I_j|}/32}} \mathcal{D}^{(j-1)}_{I_j}(x_{I_j}).$$

We can easily upper bound Z as follows.

$$Z < \binom{|I_{j-1}|}{|I_j|} \cdot t^{|I_j|} \cdot \frac{(4t)^{-|I_j|}}{32} = \binom{|I_{j-1}|}{|I_j|} 2^{-2|I_j|-5}.$$

Invariant 2.8(2) relates $\mathcal{D}^{(j-1)}$ and $\widehat{\mathcal{D}}^{(j-1)}$, which lets us lower bound Z.

$$Z \geq \frac{1}{4} \sum_{\substack{I_j \subseteq I_{j-1} : \\ |I_j| = k^{1-j/r}}} \sum_{\substack{x_{I_j} \in [t]^{|I_j|} : \\ \mathcal{D}^{(j-1)}_{I_j}(x_{I_j}) < (4t)^{-|I_j|}/32}} \widehat{\mathcal{D}}^{(j-1)}_{I_j}(x_{I_j}) \qquad \text{(Invariant 2.8(2).)}$$

By definition, $\widehat{\mathcal{D}}^{(j-1)}$ is a convex combination of the $\mathcal{D}^{(j)}[m_j]$ distributions, weighted according to $\mu_j(\cdot)$. Hence, the expression above is lower bounded by

$$\geq \frac{1}{4} \sum_{\substack{I_j \subseteq I_{j-1} : \\ |I_j| = k^{1-j/r}}} \sum_{\substack{x_{I_j} \in [t]^{|I_j|} : \\ \mathcal{D}^{(j-1)}_{I_j}(x_{I_j}) < (4t)^{-|I_j|}/32}} \sum_{m_j \in \mathcal{M}''_j} \mu_j(m_j) \cdot \mathcal{D}^{(j)}_{I_j}[m_j](x_{I_j})$$

$$\geq \frac{1}{4} \sum_{m_j \in \mathcal{M}''_j} \mu_j(m_j) \sum_{\substack{I_j \subseteq J_0 : \\ |I_j| = k^{1-j/r}}} \sum_{\substack{x_{I_j} \in [t]^{|I_j|} : \\ \mathcal{D}^{(j-1)}_{I_j}(x_{I_j}) < (4t)^{-|I_j|}/32}} \mathcal{D}^{(j)}_{I_j}[m_j](x_{I_j})$$

(Rearrange sums.)

By definition, for every $m_j \in \mathcal{M}''_j$ and every choice of $I_j \subseteq J_0$, part (1) of the lemma is violated. Continuing with the inequalities,

$$> \frac{1}{4} \sum_{m_j \in \mathcal{M}''_j} \mu_j(m_j) \cdot \binom{|J_0|}{|I_j|} \cdot \frac{1}{2}$$

$$> \frac{1}{32} \binom{|I_{j-1}|/2}{|I_j|}. \qquad \text{(Because } \mu_j(\mathcal{M}''_j) > 1/4.\text{)}$$

This contradicts the upper bound on Z whenever $k^{1/r}$ is at least some sufficiently large constant. □

The receiver of m_j constructs a new distribution $\widehat{\mathcal{D}}^{(j)}$ in two steps. After fixing I_j, we construct $\widetilde{\mathcal{D}}^{(j)}$ by combining $\mathcal{D}^{(j-1)}$ and $\mathcal{D}^{(j)}$, filtering out some points in the space whose probability mass is too low. We then construct $\widehat{\mathcal{D}}^{(j)}$ from $\widetilde{\mathcal{D}}^{(j)}$ and $\mathcal{D}^{(j-1)}$ by filtering out points that occur under $\widetilde{\mathcal{D}}^{(j)}$ with substantially larger probability than they do under $\mathcal{D}^{(j-1)}$.

Formally, suppose Invariant 2.8 holds for $j-1$. For each message $m_j \in \mathcal{M}_j$ (from Lemma 2.10), let I_j be selected to satisfy both properties of Lemma 2.10. Define the

probability mass of a vector $x \in [t]^k$ under $\widetilde{\mathcal{D}}^{(j)}$ as follows:

$$\widetilde{\mathcal{D}}^{(j)}(x) = \begin{cases} 0, & \text{if } \mathcal{D}_{I_j}^{(j-1)}(x_{I_j}) < \frac{(4t)^{-|I_j|}}{32}; \\ \frac{\mathcal{D}_{I_j}^{(j)}(x_{I_j})}{\beta_1} \cdot \frac{\mathcal{D}^{(j-1)}(x)}{\mathcal{D}_{I_j}^{(j-1)}(x_{I_j})}, & \text{otherwise.} \end{cases}$$

where β_1 is

$$\beta_1 = \Pr_{x_{I_j} \sim \mathcal{D}_{I_j}^{(j)}} \left[\mathcal{D}_{I_j}^{(j-1)}(x_{I_j}) \geq \frac{(4t)^{-|I_j|}}{32} \right].$$

In other words, we discard a $1 - \beta_1$ fraction of the distribution $\mathcal{D}^{(j)}$, but ignoring this effect, the projection of $\widetilde{\mathcal{D}}^{(j)}$ onto I_j has the same distribution as $\mathcal{D}^{(j)}$ onto I_j, and conditioned on the value of x_{I_j}, the distribution $\widetilde{\mathcal{D}}^{(j)}$ (projected onto $[k] \backslash I_j$) is identical to $\mathcal{D}^{(j-1)}$. We derive $\widehat{\mathcal{D}}^{(j)}$ from $\widetilde{\mathcal{D}}^{(j)}$ with a similar transformation.

$$\widehat{\mathcal{D}}^{(j)}(x) = \begin{cases} 0, & \text{if } \frac{\widetilde{\mathcal{D}}_{I_j}^{(j)}(x_{I_j})}{\mathcal{D}_{I_j}^{(j-1)}(x_{I_j})} > 2^{\gamma_j}; \\ \frac{\widetilde{\mathcal{D}}_{I_j}^{(j)}(x_{I_j})}{\beta_2} \cdot \frac{\mathcal{D}^{(j-1)}(x)}{\mathcal{D}_{I_j}^{(j-1)}(x_{I_j})}, & \text{otherwise.} \end{cases}$$

where β_2 and γ_j are defined to be

$$\begin{aligned} \beta_2 &= \Pr_{x_{I_j} \sim \widetilde{\mathcal{D}}_{I_j}^{(j)}} \left[\frac{\widetilde{\mathcal{D}}_{I_j}^{(j)}(x_{I_j})}{\mathcal{D}_{I_j}^{(j-1)}(x_{I_j})} \leq 2^{\gamma_j} \right], \\ \gamma_j &= \sum_{u=1}^{j} l_u \left(\frac{16}{k^{1/r}} \right)^{j-u+1} + (16 \cdot 22^{j-1} + 6)|I_j| + 6 \\ &\leq \sum_{u=1}^{j} l_u \left(\frac{16}{k^{1/r}} \right)^{j-u+1} + 22^j |I_j| + 6. \end{aligned}$$

The proofs of Lemmas 2.11 and 2.12 use several simple observations about $\widetilde{\mathcal{D}}^{(j)}$ and $\widehat{\mathcal{D}}^{(j)}$:

1. Lemma 2.10(1) states that $\beta_1 \geq 1/2$. Lemma 2.10(2) lower bounds the entropy of $\mathcal{D}_{I_j}^{(j)}$. We apply Lemma 2.6 to $\mathcal{D}_{I_j}^{(j)}$ and $\widetilde{\mathcal{D}}_{I_j}^{(j)}$ (taking the roles of p and q, respectively) with parameter $\alpha = 1/2 \leq \beta_1$, and obtain the following lower

bound on the entropy of $\widetilde{\mathcal{D}}^{(j)}_{I_j}$.

$$\mathrm{H}(\widetilde{\mathcal{D}}^{(j)}_{I_j}) \geq \left(cE - 8\sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{1-(u-1)/r}} - 8 \cdot 22^{j-1} - 2 \right) |I_j|.$$

2. We can then apply Lemma 2.7 to $\mathcal{D}^{(j-1)}_{I_j}$ and $\widetilde{\mathcal{D}}^{(j)}_{I_j}$ (taking the roles of p and q, respectively) with parameters

$$\begin{aligned} g_1 &= 8\sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{(j-u+1)/r}} + (8 \cdot 22^{j-1} + 2)|I_j|, \\ g_2 &= 2|I_j| + 5, \\ \text{and } \alpha &= 1/2. \end{aligned}$$

Since $g_1/\alpha + g_2 - (1-\alpha)\log(1-\alpha)/\alpha = \gamma_j$, we conclude that $\beta_2 \geq 1 - \alpha = 1/2$. Thus, for each value $x_{I_j} \in \mathrm{supp}(\widehat{\mathcal{D}}^{(j)}_{I_j})$,

$$\widehat{\mathcal{D}}^{(j)}_{I_j}(x_{I_j}) = \frac{\widetilde{\mathcal{D}}^{(j)}_{I_j}(x_{I_j})}{\beta_2} = \frac{\mathcal{D}^{(j)}_{I_j}(x_{I_j})}{\beta_1 \beta_2} \leq 4\mathcal{D}^{(j)}_{I_j}(x_{I_j}). \tag{2.2}$$

Lemma 2.11 completes the inductive step by lower bounding the entropy of $\widehat{\mathcal{D}}^{(j)}_{I'}$ for every nonempty subset $I' \subseteq I_j$. To put it another way, it ensures that the coordinates in I_j remain almost completely unknown to both parties.

Lemma 2.11. *Fix $j \in [1, r]$ and suppose Invariant 2.8 holds for $j-1$. Then, for each message $m_j \in \mathcal{M}_j$ (from Lemma 2.10), Invariant 2.8 also holds for j.*

Proof. Due to Lemma 2.10 and Eqn. (2.2), the first two properties of Invariant 2.8 are satisfied. For each nonempty subset $I' \subseteq I_j$, the third property of Invariant 2.8 can be derived from the second property of Lemma 2.10 and an application of Lemma 2.6 to $\mathcal{D}^{(j)}_{I'}$ and $\widehat{\mathcal{D}}^{(j)}_{I'}$ (taking the roles of p and q, respectively) with parameter $\alpha = 1/4$ as follows.

$$\begin{aligned} \mathrm{H}(\widehat{\mathcal{D}}^{(j)}_{I'}) &\geq \left(cE - 16\sum_{u=1}^{j} \frac{16^{j-u} l_u}{k^{1-(u-1)/r}} - 16 \cdot 22^{j-1} - 4 \right) |I'| \\ &\geq \left(cE - \sum_{u=1}^{j} \frac{16^{j-u+1} l_u}{k^{1-(u-1)/r}} - 22^j \right) |I'|. \end{aligned}$$

□

Aside from maintaining Invariant 2.8 round by round, another important part of our proof is to compute the error probability. Lemma 2.12 shows how the error probabilities of two consecutive rounds are related after our modification to the protocol. More importantly, it also illustrates the reason to bound the *pointwise* ratio between $\widetilde{\mathcal{D}}^{(j)}_{I_j}$ and $\mathcal{D}^{(j-1)}_{I_j}$.

Lemma 2.12. *Fix a round $j \in [1, r]$ and suppose Invariant 2.8 holds for $j-1$. Fix any specific message $m_j \in \mathcal{M}_j$ (from Lemma 2.10). Define p to be the probability of error, when the protocol begins after round j with the inputs drawn from $\mathcal{D}^{(j)}$ and $\widehat{\mathcal{D}}^{(j)}$, respectively. Then the probability of error is at least $2^{-\gamma_j-1}p$ when the inputs are instead drawn from $\mathcal{D}^{(j)}$ and $\mathcal{D}^{(j-1)}$, respectively.*

Proof. From the definition of $\widehat{\mathcal{D}}^{(j)}$, for each value $x \in \mathrm{supp}(\widehat{\mathcal{D}}^{(j)})$, we have

$$\frac{\widehat{\mathcal{D}}^{(j)}(x)}{\mathcal{D}^{(j-1)}(x)} \;=\; \frac{\widetilde{\mathcal{D}}^{(j)}_{I_j}(x_{I_j})}{\beta_2 \mathcal{D}^{(j-1)}_{I_j}(x_{I_j})} \;\le\; \frac{2^{\gamma_j}}{\beta_2} \;\le\; 2^{\gamma_j+1}. \tag{2.3}$$

This essentially concludes the proof. □

Finally, with all lemmas proved above, we have reached the point to calculate the initial error probability.

Lemma 2.13. *Recall that $c = 1/2, c' = c/100$. Fix any $r \in [1, (\log k)/6]$ and $E \ge 100k^{1-1/r}/c$. Suppose the initial input vectors are drawn independently and uniformly from $[t]^k$, where $t = 2^{cE}$. Then the error probability of the* EqualityTesting *protocol, p_{err}, is greater than 2^{-E}.*

Proof. First suppose Invariant 2.8 holds for r and consider the situation after the final round, where the inputs are drawn from $\mathcal{D}^{(r)}$ and $\widehat{\mathcal{D}}^{(r)}$, respectively. Notice that I_r is a singleton set, so the entropy of $\widehat{\mathcal{D}}^{(r)}_{I_r}$ can be lower bounded as follows.

$$\begin{aligned}
\mathrm{H}(\widehat{\mathcal{D}}^{(r)}_{I_r}) &\ge cE - \sum_{u=1}^{r} \frac{16^{r-u+1} l_u}{k^{1-(u-1)/r}} - 22^r && \text{Invariant 2.8(3).} \\
&= cE - \frac{16}{k^{1/r}} \sum_{u=1}^{r} l_u \left(\frac{16}{k^{1/r}}\right)^{r-u} - 22^r \\
&\ge cE - \frac{16}{k^{1/r}} \sum_{u=1}^{r} l_u - 22k^{1-1/r} && k^{1/r} \ge 2^6 \text{ due to } r \le (\log k)/6. \\
&\ge cE - 16c'E - 22k^{1-1/r} \;>\; \frac{cE}{2}. && \text{Because } \textstyle\sum_{u=1}^{r} l_u \le c'Ek^{1/r}.
\end{aligned}$$

From the lower bound on the entropy of $\widehat{\mathcal{D}}^{(r)}_{I_r}$, we can easily show that there exists no value x_{I_r} such that $\widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) = \alpha > 3/4$. If there *were* such a value, then the entropy of $\widehat{\mathcal{D}}^{(r)}_{I_r}$ can also be upper bounded as

$$\mathrm{H}(\widehat{\mathcal{D}}^{(r)}_{I_r}) \;\le\; \alpha \log \frac{1}{\alpha} + (1-\alpha)\log\frac{t}{1-\alpha} \;<\; \frac{cE}{4} + \alpha\log\frac{1}{\alpha} + (1-\alpha)\log\frac{1}{1-\alpha} \;<\; \frac{cE}{2},$$

contradicting the lower bound on $\mathrm{H}(\widehat{\mathcal{D}}^{(r)}_{I_r})$.

After all r rounds of communication, the receiver of the last message has to make the decision on I_r depending only on his own input on I_r. Let $\mathcal{X}_0 \subseteq [t]$ be the subset of values x_{I_r} such that the protocol outputs "not equal" on I_r upon seeing the input x_{I_r} after r rounds of communication, $\mathcal{X}_1 = [t] \setminus \mathcal{X}_0$, and $\beta = \widehat{\mathcal{D}}^{(r)}_{I_r}(\mathcal{X}_0)$. Then, the final error probability is at least

$$\begin{aligned}
&\sum_{x_{I_r}\in\mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})\mathcal{D}^{(r)}_{I_r}(x_{I_r}) + \sum_{x_{I_r}\in\mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})\left(1-\mathcal{D}^{(r)}_{I_r}(x_{I_r})\right)\\
&= \sum_{x_{I_r}\in\mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})\mathcal{D}^{(r)}_{I_r}(x_{I_r}) + \sum_{x_{I_r}\in\mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) \sum_{x'_{I_r}\neq x_{I_r}} \mathcal{D}^{(r)}_{I_r}(x'_{I_r})\\
&\ge \frac{1}{4}\sum_{x_{I_r}\in\mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})^2 + \frac{1}{4}\sum_{x_{I_r}\in\mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) \sum_{x'_{I_r}\neq x_{I_r}} \widehat{\mathcal{D}}^{(r)}_{I_r}(x'_{I_r}) && \text{(Invariant 2.8(2).)}\\
&= \frac{1}{4}\sum_{x_{I_r}\in\mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})^2 + \frac{1}{4}\sum_{x_{I_r}\in\mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})\left(1-\widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})\right)\\
&\ge \frac{1}{4}\sum_{x_{I_r}\in\mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})^2 + \frac{1}{16}\sum_{x_{I_r}\in\mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) && \text{(Because } \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) \le 3/4.\text{)}\\
&\ge \frac{\beta^2}{4t} + \frac{1-\beta}{16} \;\ge\; \frac{1}{4t}. && \text{(Convexity of } x^2.\text{)}
\end{aligned}$$

This result also meets the simple intuition that when the inputs to the two parties are almost uniformly random and no communication is allowed, the best strategy would be guessing "not equal" regardless of the actual input.

Finally, we are ready to transfer the error probability back round by round. From Lemma 2.10 through Lemma 2.12, the error probability w.r.t. $\mathcal{D}^{(j)}$ and $\widehat{\mathcal{D}}^{(j)}$ differs from the error probability w.r.t. $\mathcal{D}^{(j-1)}$ and $\widehat{\mathcal{D}}^{(j-1)}$ by at most a $4 \cdot 2^{\gamma_j+1} = 2^{\gamma_j+3}$ factor. In particular, Lemma 2.10 and Lemma 2.11 say that the jth message m_j satisfies Invariant 2.8 at index j with probability at least $1/4$, provided Invariant 2.8 holds for $j-1$, and Lemma 2.12 says the error probabilities under the two measures differ by a 2^{γ_j+1} factor for any such m_j. Repeating this for each $j \in [1, r]$, we conclude

that the initial error probability p_{err} is lower bounded by

$$p_{\text{err}} \geq \frac{1}{4t} \cdot \exp\left(-3r - \sum_{j=1}^{r} \gamma_j\right) = \exp\left(-cE - 2 - 3r - \sum_{j=1}^{r} \gamma_j\right) > 2^{-E},$$

since

$$\begin{aligned}
&cE + 2 + 3r + \sum_{j=1}^{r} \gamma_j \\
&\leq cE + 2 + 3r + 6r + \sum_{j=1}^{r} \sum_{u=1}^{j} l_u \left(\frac{16}{k^{1/r}}\right)^{j-u+1} + \sum_{j=1}^{r} 22^j |I_j| \\
&\leq cE + 11r + \sum_{u=1}^{r} \frac{16 l_u}{k^{1/r}} \sum_{j=u}^{r} \left(\frac{16}{k^{1/r}}\right)^{j-u} + 22k^{1-1/r} \sum_{j=1}^{r} \left(\frac{22}{k^{1/r}}\right)^{j-1} && \text{(Rearrange sums.)} \\
&\leq cE + 11r + \frac{32}{k^{1/r}} \sum_{u=1}^{r} l_u + 44k^{1-1/r} && (k^{1/r} \geq 2^6 \text{ since } r \leq (\log k)/6.) \\
&\leq cE + \frac{11cE}{100} + \frac{32cE}{100} + \frac{44cE}{100} \;<\; E. && (\text{Because } \textstyle\sum_{u=1}^{r} l_u \leq c' E k^{1/r}.)
\end{aligned}$$

□

Proof of Theorem 2.4. Lemma 2.13 actually shows that given integers $k \geq 1$ and $r \leq (\log k)/6$, any r-round deterministic protocol for EqualityTesting on vectors of length k that has distributional error probability $p_{\text{err}} = 2^{-E}$ with respect to the uniform input distribution on $[t]^k$, where $t = 2^{cE}$, requires at least $\Omega(Ek^{1/r})$ bits of communication. Notice that the additional assumption $E \geq 100k^{1-1/r}/c$ always makes sense since there is a trivial $\Omega(k)$ lower bound on the communication complexity of EqualityTesting, regardless of r. Thus, Theorem 2.4 follows directly from Yao's minimax principle. □

2.3.3 A Lower Bound on ExistsEqual

The proof of Theorem 2.5 is almost the same as that of Theorem 2.4, except for the final step, namely Lemma 2.13, in which we first compute the final error probability after all r rounds of communication and then transfer it backward round by round using Lemma 2.12. The problem with applying the same argument to ExistsEqual protocols is that the receiver of the last message may be able to announce the correct answer, even though it knows little information about the inputs on the single coordinate I_r.

In order to prove Theorem 2.5, first notice that Lemma 2.9 through Lemma 2.12 also hold perfectly well for ExistsEqual protocols as no modification is required in their proofs. Therefore, it is sufficient to prove the following Lemma 2.14, which is an analog of Lemma 2.13 for ExistsEqual. It is based mainly on Markov's inequality.

Lemma 2.14. *Recall that $c = 1/2, c' = c/100$. Consider an execution of a deterministic r-round ExistsEqual protocol, $r \in [1, (\log k)/6]$, on input vectors drawn independently and uniformly from $[t]^k$, where $t = 2^{cE}$. Here $E \geq 100k^{1-1/r}/c$ if $r > 1$ and $E \geq (100 \log k)/c$ otherwise. Then the protocol errs with probability $p_{\text{err}} > 2^{-E}$.*

Proof. Similarly to the proof of Lemma 2.13, we first consider the situation after the final round. In the ExistsEqual protocol, the receiver of the last message can make the decision depending on every coordinate of his own input. Let $\mathcal{X}_0 \subseteq [t]^k$ be the subset of values x such that the protocol outputs "no" upon seeing the input x after r rounds of communication, $\mathcal{X}_1 = [t]^k \setminus \mathcal{X}_0$. Then, the final error probability is at least

$$\sum_{x \in \mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}(x) \mathcal{D}^{(r)}_{I_r}(x_{I_r}) + \sum_{x \in \mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}(x) \left(1 - \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y)\right),$$

where $\mathcal{N}(x) = \{y \in [t]^k \mid \text{there exists some } i \in [k] \text{ such that } x_i = y_i\}$ is the subset of input vectors that agree with x on at least one coordinate.

The main difficulty here is to lower bound $1 - \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y)$, which is potentially quite small. Consider the following summation Z_0 over *all* transcripts $m_1, \ldots, m_r$ in which $m_j \in \mathcal{M}_j$ (from Lemma 2.10), where the set $\mathcal{M}_j$ depends on $m_1, \ldots, m_{j-1}$:

$$Z_0 = \sum_{m_1 \in \mathcal{M}_1} \mu_1(m_1) \sum_{m_2 \in \mathcal{M}_2} \mu_2(m_2) \cdots \sum_{m_r \in \mathcal{M}_r} \mu_r(m_r) \sum_{x \in [t]^k} \widehat{\mathcal{D}}^{(r)}(x) \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y).$$

From the proof of Lemma 2.12 (Eqn. (2.3)), we can upper bound Z_0 as follows.

$$Z_0 \leq \sum_{m_1 \in \mathcal{M}_1} \mu_1(m_1) \cdots \sum_{m_r \in \mathcal{M}_r} \mu_r(m_r) \sum_{\substack{x \in [t]^k, \\ y \in \mathcal{N}(x)}} 2^{\gamma_r + 1} \mathcal{D}^{(r-1)}(x) \mathcal{D}^{(r)}(y)$$

Notice that γ_r and $\mathcal{D}^{(r-1)}$ are independent of the choice of m_r, hence by rearranging sums, this is equal to

$$= \sum_{m_1 \in \mathcal{M}_1} \mu_1(m_1) \cdots \sum_{m_{r-1} \in \mathcal{M}_{r-1}} \mu_{r-1}(m_{r-1}) \sum_{\substack{x \in [t]^k, \\ y \in \mathcal{N}(x)}} 2^{\gamma_r + 1} \mathcal{D}^{(r-1)}(x) \sum_{m_r \in \mathcal{M}_r} \mu_r(m_r) \mathcal{D}^{(r)}(y)$$

By definition, $\widehat{\mathcal{D}}^{(r-1)}$ is a convex combination of the $\mathcal{D}^{(r)}[m_r]$ distributions, weighted according to $\mu_r(\cdot)$. Hence, the expression above is upper bounded by

$$\leq \sum_{m_1 \in \mathcal{M}_1} \mu_1(m_1) \cdots \sum_{m_{r-1} \in \mathcal{M}_{r-1}} \mu_{r-1}(m_{r-1}) \sum_{\substack{x \in [t]^k, \\ y \in \mathcal{N}(x)}} 2^{\gamma_r + 1} \mathcal{D}^{(r-1)}(x) \widehat{\mathcal{D}}^{(r-1)}(y)$$

By the symmetry of x and y, this is equal to

$$= \sum_{m_1 \in \mathcal{M}_1} \mu_1(m_1) \cdots \sum_{m_{r-1} \in \mathcal{M}_{r-1}} \mu_{r-1}(m_{r-1}) \sum_{\substack{x \in [t]^k, \\ y \in \mathcal{N}(x)}} 2^{\gamma_r + 1} \widehat{\mathcal{D}}^{(r-1)}(x) \mathcal{D}^{(r-1)}(y)$$

We repeat the same argument for rounds $r-1$ down to 1, upper bounding Z_0 by

$$\leq \exp\left(r + \sum_{j=1}^{r} \gamma_j\right) \sum_{\substack{x \in [t]^k, \\ y \in N(x)}} \widehat{\mathcal{D}}^{(0)}(x) \mathcal{D}^{(0)}(y)$$
$$\leq \exp\left(r + \sum_{j=1}^{r} \gamma_j\right) \frac{k}{t}$$

The last inequality above follows from a union bound since, under the initial distributions $\widehat{\mathcal{D}}^{(0)}, \mathcal{D}^{(0)}$, each of the k coordinates is equal with probability $1/t$. Recall that $E \geq 100k^{1-1/r}/c$ when $r > 1$ and $E \geq (100 \log k)/c$ otherwise. Hence, using the same argument as that in the proof of Lemma 2.13, we can further bound this as

$$\leq 2^{0.83cE} \cdot 2^{0.02cE} 2^{-cE} \;=\; 2^{-0.15cE},$$

since

$$r + \sum_{j=1}^{r} \gamma_j \leq 7r + \sum_{j=1}^{r} \sum_{u=1}^{j} l_u \left(\frac{16}{k^{1/r}}\right)^{j-u+1} + \sum_{j=1}^{r} 22^j |I_j| \leq \frac{7cE}{100} + \frac{32cE}{100} + \frac{44cE}{100} = \frac{83cE}{100},$$

and $k \leq (cE/100)^{r/(r-1)} \leq (cE/100)^2 \leq 2^{0.02cE}$ when $r > 1$ and $k \leq 2^{0.01cE}$ otherwise.

Now fix a round j and a particular history $(m_1, \ldots, m_j)$ up to round j such that $m_{j'} \in \mathcal{M}_{j'}$ holds for every $j' \leq j$. Define Z_j as follows.

$$Z_j = \sum_{m_{j+1} \in \mathcal{M}_{j+1}} \mu_{j+1}(m_{j+1}) \cdots \sum_{m_r \in \mathcal{M}_r} \mu_r(m_r) \sum_{x \in [t]^k} \widehat{\mathcal{D}}^{(r)}(x) \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y).$$

By Markov's inequality, there exists a subset of messages $\widehat{\mathcal{M}}_1 \subseteq \mathcal{M}_1$ with $\mu_1(\widehat{\mathcal{M}}_1) \geq \mu_1(\mathcal{M}_1)/2 \geq 1/8$ such that each message $m_1 \in \widehat{\mathcal{M}}_1$ satisfies $Z_1 \leq 2Z_0/\mu_1(\mathcal{M}_1) \leq 8Z_0$ since $\mu_1(\mathcal{M}_1) \geq 1/4$ from Lemma 2.10. Similarly, conditioned on any specific $m_1 \in \widehat{\mathcal{M}}_1$, by Markov's inequality, there exists a subset of messages $\widehat{\mathcal{M}}_2 \subseteq \mathcal{M}_2$ with $\mu_2(\widehat{\mathcal{M}}_2) \geq \mu_2(\mathcal{M}_2)/2 \geq 1/8$ such that each message $m_2 \in \widehat{\mathcal{M}}_2$ satisfies $Z_2 \leq 2Z_1/\mu_2(\mathcal{M}_2) \leq 8^2 Z_0$. In general, conditioned on any specific partial transcript $m_1, \ldots, m_{j-1}$ such that $m_{j'} \in \widehat{\mathcal{M}}_{j'}$ holds for every $j' < j$, there exists a subset of messages $\widehat{\mathcal{M}}_j \subseteq \mathcal{M}_j$ with $\mu_j(\widehat{\mathcal{M}}_j) \geq \mu_j(\mathcal{M}_j)/2 \geq 1/8$ such that each message $m_j \in \widehat{\mathcal{M}}_j$ satisfies $Z_j \leq 8^j Z_j$.

After repeating the same argument r times, we get $\widehat{\mathcal{M}}_1, \ldots, \widehat{\mathcal{M}}_r$ in sequence. For any sampled transcript $m_1, \ldots, m_r$ such that $m_j \in \widehat{\mathcal{M}}_j$ for all $j \leq r$, we have

$$Z_r \leq 8^r Z_0 \leq 2^{3r} \cdot 2^{-0.15cE} \leq 2^{-0.12cE} \leq \frac{1}{4},$$

as $r \leq cE/100$ and $cE \geq 100$. Further, one more application of Markov's inequality shows that there exists a subset of values $\mathcal{X}' \subseteq [t]^k$ with $\widehat{\mathcal{D}}^{(r)}(\mathcal{X}') = \alpha \geq 1/2$ such that $\sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y) \leq 1/2$ holds for every $x \in \mathcal{X}'$.

As a result, we can then lower bound the final error probability as follows, where $\beta = \widehat{\mathcal{D}}^{(r)}(\mathcal{X}_0 \cap \mathcal{X}')$.

$$\begin{aligned}
&\sum_{x \in \mathcal{X}_0} \widehat{\mathcal{D}}^{(r)}(x) \mathcal{D}^{(r)}_{I_r}(x_{I_r}) + \sum_{x \in \mathcal{X}_1} \widehat{\mathcal{D}}^{(r)}(x) \left(1 - \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y)\right) \\
&\geq \sum_{x \in (\mathcal{X}_0 \cap \mathcal{X}')} \widehat{\mathcal{D}}^{(r)}(x) \mathcal{D}^{(r)}_{I_r}(x_{I_r}) + \sum_{x \in (\mathcal{X}_1 \cap \mathcal{X}')} \widehat{\mathcal{D}}^{(r)}(x) \left(1 - \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y)\right) \\
&\geq \frac{1}{4} \sum_{x \in (\mathcal{X}_0 \cap \mathcal{X}')} \widehat{\mathcal{D}}^{(r)}(x) \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) + \sum_{x \in (\mathcal{X}_1 \cap \mathcal{X}')} \widehat{\mathcal{D}}^{(r)}(x) \left(1 - \sum_{y \in \mathcal{N}(x)} \mathcal{D}^{(r)}(y)\right) \\
&\qquad\qquad \text{(Invariant 2.8(2).)} \\
&\geq \frac{1}{4} \sum_{x \in (\mathcal{X}_0 \cap \mathcal{X}')} \widehat{\mathcal{D}}^{(r)}(x) \widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r}) + \frac{1}{2} \sum_{x \in (\mathcal{X}_1 \cap \mathcal{X}')} \widehat{\mathcal{D}}^{(r)}(x) \qquad \text{(Defn. of } \mathcal{X}'.)
\end{aligned}$$

In order to minimize the above expression, we can now assume without loss of generality that the partition between $\mathcal{X}_0 \cap \mathcal{X}'$ and $\mathcal{X}_1 \cap \mathcal{X}'$ depends solely on x_{I_r} as only the relative magnitude of $\widehat{\mathcal{D}}^{(r)}_{I_r}(x_{I_r})/4$ and $1/2$ matters. Continuing,

$$\geq \frac{\beta^2}{4t} + \frac{\alpha - \beta}{2} \geq \frac{\alpha^2}{4t} \geq \frac{1}{16t}. \qquad \text{(Convexity of } x^2\text{.)}$$

Finally, we are ready to transfer the error probability back in exactly the same manner as we did in the proof of Lemma 2.13. Using a similar argument, the existence of $\widehat{\mathcal{M}}_j$ guarantees that

$$p_{\text{err}} \geq \frac{1}{16t} \cdot \exp\left(-4r - \sum_{j=1}^{r} \gamma_j\right) = \exp\left(-cE - 4 - 4r - \sum_{j=1}^{r} \gamma_j\right) > 2^{-E},$$

since

$$cE + 4 + 4r + \sum_{j=1}^{r} \gamma_j \leq cE + \frac{14cE}{100} + \frac{32cE}{100} + \frac{44cE}{100} < E.$$

□

Proof of Theorem 2.5. Similarly to the proof of Theorem 2.4, Theorem 2.5 follows from Lemma 2.14 and a direct application of Yao's minimax principle. □

2.4 Upper Bounds on EqualityTesting and ExistsEqual

In this section, we prove upper bounds on both EqualityTesting and ExistsEqual. We first give a $(\log^*(k/E) + r)$-round EqualityTesting protocol (Theorem 2.15) that uses $O(k + rEk^{1/r})$ bits of communication and errs with probability at most $p_{\text{err}} = 2^{-E}$. The $\log^*(k/E)$ term cannot be completely eliminated, due to the lower bounds of [130, 23]. Our lower bound implies that when $E \geq k$ (so $\log^*(k/E) = 0$), the second term is optimal up to a factor of r.

A natural goal is to achieve optimal communication $\Theta(k + E)$ and minimize the number of rounds subject to that constraint. When $E \geq k$ our lower bound says $r = \Omega(\log k)$, but in this case the algorithm of Theorem 2.15 only achieves $O(E \log k)$ communication. Theorems 2.16 and 2.17 illustrate two ways to shave off this factor of r. Theorem 2.16 applies to the easier ExistsEqual problem, and Theorem 2.17 applies to the general EqualityTesting problem, but blows up the round complexity to $\log^*(k/E) + O(r)$.

Theorem 2.15. *There exists a* $(\log^*(k/E)+r)$*-round randomized protocol for* EqualityTesting *on vectors of length* k *that errs with probability* $p_{\text{err}} = 2^{-E}$*, using* $O(k + rEk^{1/r})$ *bits of communication.*

Theorem 2.16. *There exists a* $(\log^*(k/E)+r)$*-round randomized protocol for* ExistsEqual *on vectors of length* k *that errs with probability* $p_{\text{err}} = 2^{-E}$*, using* $O(k+Ek^{1/r})$ *bits of communication.*

Theorem 2.17. *There exists a* $(\log^*(k/E) + O(r))$*-round randomized protocol for* EqualityTesting *on vectors of length* k *that errs with probability* $p_{\text{err}} = 2^{-E}$*, using* $O(k + Ek^{1/r})$ *bits of communication.*

Theorem 2.18. *There exists a* $(\log^*(k/E)+r)$*-round randomized protocol for* EqualityTesting *on vectors of length* k *that errs with probability* $p_{\text{err}} = 2^{-E}$*, using* $O(k + Ek^{1/r}\log r + Er\log r)$ *bits of communication.*

Remark 2.19. *The* $\log^*(k/E)$ *terms in the round complexity of Theorems 2.15–2.17 are not absolute. They can each be replaced with* $\max\{0,\, \log^*(k/E) - \log^*(C)\}$*, at the cost of increasing the communication by* $O(Ck)$*.*

2.4.1 Overview and Preliminaries

We start by giving a generic protocol for EqualityTesting. The protocol uses a simple subroutine for ExistsEqual/EqualityTesting when $k = 1$. Suppose Alice and Bob hold $x, y \in U = \{0,1\}^l$, respectively. Alice picks a random $w \in \{0,1\}^l$ from the shared random source and sends Bob $\check{x} = \langle x, w\rangle \bmod 2$, where $\langle\cdot,\cdot\rangle$ is the inner product operator. Bob computes $\check{y} = \langle y, w\rangle \bmod 2$ and declares "$x = y$" iff $\check{x} = \check{y}$. Clearly, Bob never errs if $x = y$; it is straightforward to show that the probability of error is exactly $1/2$ when $x \neq y$. We call this protocol an *inner product test* and $\check{x}, \check{y}$ *test bits*. A *b-bit inner product test* on x and y refers to b independent inner product tests on x and y.

The entire protocol is divided into several phases. Before phase j, $j \geq 1$, Alice and Bob agree on a subset I_{j-1} of coordinates on which all previous inner product tests have passed. In other words, they have not yet witnessed that any of the coordinates in I_{j-1} are not equal. Each coordinate $i \in I_{j-1}$ represents either an actual equality ($x_i = y_i$), or a *false positive* ($x_i \neq y_i$). At the beginning of the protocol, $I_0 = [k]$. In phase j, we perform l_j independent inner product tests on each coordinate in I_{j-1} and let $I_j \subseteq I_{j-1}$ be the remaining coordinates that pass all their respective inner product tests. Notice that each coordinate in I_{j-1} corresponding to equality will always pass

all the tests and enter I_j, while those corresponding to inequalities will only enter I_j with probability 2^{-l_j}. At the end of the protocol, we declare all coordinates in I_r *equal* and all other coordinates *not equal*.

This finishes the description of our generic protocol. Theorems 2.15–2.17 all use the framework of the generic protocol and mainly differ in the details, such as how Alice and Bob exchange their test bits, how they decide l_j, and when the protocol terminates.

2.4.1.1 A protocol for exchanging test bits

For EqualityTesting, it is possible that a constant fraction of the coordinates are actually equalities, which makes $|I_j| = \Theta(k)$ for every j. The naive implementation explicitly exchanges all $l_j|I_{j-1}|$ test bits and uses $\Omega(kE)$ bits of communication in total. All the test bits corresponding to equalities are "wasted" in a sense.

For our application, it is important that the communication volume that Alice and Bob use to exchange their test bits in phase j be proportional to the number of false positives in I_{j-1}, instead of the size of I_{j-1}. We will use a slightly improved version of a protocol of Feder et al. [63] for exchanging the test bits.

Imagine packing the test bits into vectors $\hat{x}, \hat{y} \in B^{|I_{j-1}|}$ where $B = \{0,1\}^{l_j}$. Lemma 2.20 shows that Alice can transmit $\hat{x}$ to Bob, at a cost that depends on an *a priori* upper bound on the Hamming distance $\mathrm{dist}(\hat{x}, \hat{y})$, i.e.,, the number of the coordinates in I_{j-1} where they differ.

Lemma 2.20 (Cf. Feder et al. [63].)**.** *Suppose Alice and Bob hold length-K vectors $x, y \in B^K$, where $B = \{0,1\}^L$. Alice can send one $O(dL + d\log(K/d))$-bit message to Bob, who generates a string $x' \in B^K$ such that the following holds. If the Hamming distance $\mathrm{dist}(x,y) \le d$ then $x = x'$; if $\mathrm{dist}(x,y) > d$ then there is no guarantee.*

Proof. Define $G = (V, E)$ to be the graph on $V = B^K$ such that $\{u, v\} \in E$ iff $\mathrm{dist}(u, v) \le 2d$. The maximum degree in G is clearly at most $\Delta = \binom{K}{2d} \cdot 2^{2Ld}$ since there are $\binom{K}{2d}$ ways to select the $2d$ indices and 2^{2Ld} ways to change the coordinates at those indices so that there are *at most* $2d$ different coordinates. Let $\phi : V \mapsto [\Delta + 1]$ be a proper $(\Delta + 1)$-coloring of G. Alice sends $\phi(x)$ to Bob, which requires $\log(\Delta + 1) = O(dL + d\log(K/d))$ bits. Every string in the radius-d ball around y (w.r.t. dist) is colored differently since they are all at distance at most $2d$, hence if $\mathrm{dist}(x, y) \le d$, Bob can reconstruct x without error. □

Corollary 2.21. *Suppose at phase j, it is guaranteed that the number of false positives in I_{j-1} is at most k_{j-1}. Then phase j can be implemented with $O(k_{j-1}l_j + k_{j-1}\log(k/k_{j-1}))$ bits in 2 rounds.*

Finally, a naive implementation of the protocol requires $2r$ rounds if the generic protocol has r phases. In fact, the protocol can be compressed into exactly r rounds in the following way. At the beginning, both parties agree that $I_0 = [k]$. Alice generates her $l_1|I_0|$ test bits $\hat{x}^{(1)}$ for phase 1 and communicates them to Bob; Bob first generates his own test bits $\hat{y}^{(1)}$ for phase 1 and determines I_1, then generates $l_2|I_1|$ test bits $\hat{y}^{(2)}$ for phase 2 and transmits both $\hat{y}^{(1)}$ and $\hat{y}^{(2)}$ to Alice. Alice computes I_1, generates $\hat{x}^{(2)}$, computes I_2, generates $\hat{x}^{(3)}$, and sends $\hat{x}^{(2)}$ and $\hat{x}^{(3)}$ to Bob, and so on. There is no asymptotic increase in the communication volume.

2.4.1.2 Reducing the number of false positives

Our protocols for EqualityTesting and ExistsEqual are divided into two parts. The goal of the first part is to reduce the number of false positives from at most k to at most E; if $E \geq k$, we can skip this part. Since the number of false positives is large in this part, we can use standard Chernoff bounds to control the number of false positives surviving each phase. The details are very similar to the upper bound in Sağlam and Tardos [130].

Theorem 2.22. *Let (x, y) be an instance of ExistsEqual with $|x| = |y| = k$. In $\log^*(k/E)$ rounds, we can reduce this to a new instance (x', y') of ExistsEqual where $|x'| = |y'| \leq E$, using $O(k)$ communication. The failure probability of this protocol is at most $2^{-(E+1)}$.*

For EqualityTesting, we can reduce the initial instance to a new instance (x', y') such that the Hamming distance $\mathrm{dist}(x', y') \leq E$, with the same round complexity, communication volume, and error probability.

Proof. We first give the protocol for ExistsEqual, then apply the necessary changes to make it work for EqualityTesting.

The protocol for ExistsEqual uses our generic protocol, and imposes a strict upper bound k_j on $|I_j|$. Whenever $|I_j|$ exceeds this upper bound, we halt the entire protocol and answer *yes* (there exists a coordinate where the input vectors are equal). We

start by setting the parameters k_j and l_j for any $j \in [1, \log^*(k/E)]$ as follows.

$$\begin{aligned} k_0 &= k, \\ k_j &= \max\left\{\frac{k}{2^{j-1}\exp^{(j)}(2)}, E\right\}, \\ l_j &= 3 + \exp^{(j-1)}(2). \end{aligned}$$

Note that it is reasonable to assume $k_j > E$ before the last phase, since whenever we find $k_j \leq E$, we can simply terminate the protocol prematurely after phase j, and our goal would be achieved.

Now suppose the input vectors share no equal coordinates. We know that $|I_{j-1}| \leq k_{j-1}$ at the beginning of phase j. The probability of any particular coordinate in I_{j-1} passing all tests in phase j is exactly $p_j = \exp(-l_j)$. Thus, the expected size of I_j is at most

$$k_{j-1}p_j = \frac{k}{2^{j-2}\exp^{(j-1)}(2)} \cdot \frac{1}{2^3 \exp^{(j)}(2)} \leq \frac{k}{2^{j+2}\exp^{(j)}(2)} \leq \frac{k_j}{8}.$$

Recall the statement of the usual Chernoff bound.

Fact 2.23 (See [58])**.** *Let $X = \sum_{i=1}^{n} X_i$, where each X_i is an i.i.d. Bernoulli random variable. Letting $\mu = \mathrm{E}[X]$, the following inequality holds for any $\delta > 0$.*

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu.$$

In our case $X_i = 1$ iff the ith coordinate in I_{j-1} survives to I_j. By linearity of expectation, $\mu \leq k_j/8$. Setting $\delta = k_j/\mu - 1 \geq 7$, we have

$$\Pr[X \geq k_j] = \Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\frac{k_j}{1+\delta}} < 0.3^{k_j} < 2^{-1.7k_j}.$$

The second to last inequality holds since $\left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\frac{1}{1+\delta}} < 0.3$ when $\delta \geq 7$.

Hence, the probability that there are at least k_j coordinates remaining after phase j is at most $2^{-1.7k_j}$, and the probability this happens in any phase is at most $\sum_j 2^{-1.7k_j} \leq 2^{-(E+1)}$. Notice that when x and y share at least one equal coordinate, the error probability of this protocol is 0 because if it fails to reduce the number of coordinates to E it (correctly) answers *yes*. The communication volume of the

protocol is asymptotic to

$$\sum_j l_j |I_{j-1}| \leq \sum_j l_j k_{j-1} = \sum_j O(k/2^j) = O(k).$$

For EqualityTesting, we use the same k_j as an upper bound on the number of false positives in I_j, instead of the size of I_j. Since the number of false positives is at most k at the beginning, we can still use the same argument to show that with the same choice of k_j and l_j, after $\log^*(k/E)$ phases, the number of false positives is at most E with error probability $2^{-(E+1)}$. By Corollary 2.21, the number of bits we need to exchange in phase j is $O(k_{j-1}l_j + k_{j-1}\log(k/k_{j-1}))$. Notice that $\log(k/k_{j-1}) = j - 2 + \exp^{(j-2)}(2) = O(l_j)$, so the total communication volume is still $O(k)$. □

In all of our protocols, we first apply Theorem 2.22 to reduce the number of coordinates (in the case of ExistsEqual) or false positives (in the case of EqualityTesting) to be at most E. This requires no communication if $E \geq k$ to begin with. Hence, with $\log^*(k/E)$ extra rounds and $O(k)$ communication, we will assume henceforth that all instances of ExistsEqual have $E \geq k$ and instances of EqualityTesting have $\text{dist}(x, y) \leq E$.

2.4.2 An $O(k + rEk^{1/r})$-bit EqualityTesting Protocol

In light of Theorem 2.22, we can assume that the input vectors to EqualityTesting are guaranteed to differ in at most $k_0 = \min\{k, E\}$ coordinates.

Theorem 2.24. *Fix any $k \geq 1$, $E \geq 1$, and $r \in [1, (\log k_0)/2]$, where $k_0 = \min\{k, E\}$. There exists a randomized protocol for EqualityTesting length-k vectors x, y with Hamming distance $\text{dist}(x, y) \leq k_0$ that uses r rounds, $O(k + rEk_0^{1/r})$ bits of communication, and errs with probability $p_{\text{err}} = 2^{-(E+1)}$.*

Proof. We begin by giving the parameters k_j and l_j.

$$k_j = k_0^{1-j/r},$$
$$l_j = 4Ek_0^{j/r-1}.$$

Now fix a phase $j \in [1, r]$ and suppose at the beginning of phase j that the number of false positives in I_{j-1} is at most k_{j-1}. By assumption this holds for $j = 1$. The

probability that at least k_j false positives survive phase j is upper bounded by

$$\begin{aligned}
\binom{k_{j-1}}{k_j} 2^{-k_j l_j} &\le \left(\frac{e k_{j-1}}{k_j}\right)^{k_j} 2^{-k_j l_j} && \text{Because } \tbinom{n}{k} \le \left(\tfrac{en}{k}\right)^k. \\
&\le 2^{2k_j \log(k_{j-1}/k_j) - k_j l_j} && e \le k_0^{1/r} = \tfrac{k_{j-1}}{k_j} \text{ due to } r \le \tfrac{\log k_0}{2}. \\
&\le 2^{-2E}. && \text{Because } \log \tfrac{k_{j-1}}{k_j} \le \tfrac{k_{j-1}}{k_j} = k_0^{1/r} \le \tfrac{l_j}{4}.
\end{aligned}$$

Thus, by a union bound, the number of false positives surviving phase j is *strictly less than* k_j, for all $j \in [1, r]$, with probability at least $1 - 2^{-(E+1)}$. In particular, there are no false positives at the end since $k_r = 1$.

Meanwhile, by Corollary 2.21, the total communication volume is $O(k + rEk_0^{1/r})$ since

$$\sum_{j=1}^{r} k_{j-1} l_j = 4rEk_0^{1/r},$$

and

$$\begin{aligned}
\sum_{j=1}^{r} k_{j-1} \log \frac{k}{k_{j-1}} &= k_0 \sum_{j=0}^{r-1} \frac{1}{k_0^{j/r}} \left(\log \frac{k}{k_0} + \log k_0^{j/r} \right) \\
&\le 2k_0 \log \frac{k}{k_0} + k_0 \sum_{j=0}^{r-1} \frac{\log k_0^{j/r}}{k_0^{j/r}} && k_0^{1/r} \ge 2^2 \text{ due to } r \le \tfrac{\log k_0}{2}. \\
&= O(k). && \text{Because } k_0^{1/r} \ge 2^2 \text{ and } k_0 \le k.
\end{aligned}$$

□

Proof of Theorem 2.15. Applying Theorem 2.22 and Theorem 2.24 in sequence, we obtain a $(\log^*(k/E) + r)$-round randomized protocol for EqualityTesting on vectors of length k that errs with probability $p_{\text{err}} = 2^{-E}$ and uses $O(k + rE \min\{k, E\}^{1/r})$ bits of communication. When $E \ge k$ the protocol is obtained directly from Theorem 2.24 and uses $O(rEk^{1/r})$ communication. When $E < k$ the communication implied by Theorems 2.22 and 2.24 is $O(k + rE^{1+1/r}) = O(k + rEk^{1/r})$.[7] □

[7]It appears as if $rE^{1+1/r}$ is an improvement over $rEk^{1/r}$ when $E < k$, but this is basically an illusion. In light of Remark 2.19, we can always dedicate $\log^*(k/E) - 2$ rounds to the first part and $r + 2$ rounds to the second part while increasing the communication by $O(k)$. When $E \ge k^{1-1/r}$, $rE^{1+1/r} = \Omega((r+2)Ek^{1/(r+2)})$, meaning there is no clear benefit to use the $rE^{1+1/r}$ expression.

2.4.3 An $O(k + Ek^{1/r})$-bit ExistsEqual Protocol

2.4.3.1 Overview of the protocol

In this section, we show that we can obtain a $(\log^*(k/E)+r)$-round, $O(k+Ek^{1/r})$-bit protocol for ExistsEqual. This matches the lower bound of Theorem 2.5, asymptotically, when $E \geq k$. Theorem 2.22 covers the first part of the protocol, so we assume without loss of generality that $E \geq k$.

Suppose the inputs x and y share no equal coordinates. Imagine writing down all the possible results of the inner product tests in a matrix A of dimension $(E+\log k)\times k$, where $A_{j,i}$ is "=" if x_i, y_i pass the jth inner product test, and "$\neq$" otherwise. By a union bound, with probability $1 - 2^{-E}$, each column contains at least one "$\neq$". Now consider the area above the first "$\neq$" in each column. The probability that this area is at least E' is, by a union bound, at most

$$\binom{E' + k - 1}{k - 1} 2^{-E'} < \exp(k \log(e(E' + k)/k) - E'). \tag{2.4}$$

For $E' = E + O(k \log(E/k)) = O(E)$, this probability is $\ll 2^{-E}$. In our analysis it suffices to consider a situation where an *adversary* can decide the contents of A, subject to the constraint that its *error budget* (the area above the curve defined by the first "$\neq$" in each column) never exceeds $E' = O(E)$. The notion of an error budget is also essential for analyzing the protocol of Section 2.4.4.

In the jth phase, $j \geq 1$, our protocol exposes the fragment of A consisting of the next l_j rows of columns in I_{j-1}. The set I_j consists of those columns without any "$\neq$" exposed so far. The *communication budget* for phase j is equal to $l_j|I_{j-1}|$. In the worst case, the first exposed value in each column of $I_{j-1} \setminus I_j$ is "$\neq$", so the adversary spends at least $l_j|I_j|$ of its *error budget* in phase j.

If we witness at least one "$\neq$" in every column, we can correctly declare there does not exist an equal coordinate and answer *no*. Otherwise, if the adversary has not exceeded his error budget but there is some column without any "$\neq$", we answer *yes*. If the adversary ever exhausts his error budget, we terminate the protocol and answer *yes*. Recall that the notion of an error budget tacitly assumed that x and y differ in all coordinates. If they do not, the protocol always answers correctly, whether it halts prematurely or not. The probability that the error budget is exhausted when x and y differ in all coordinates (a false positive) is $\ll 2^{-E}$, according to Eqn. (2.4).

2.4.3.2 Analysis

In this section we give a formal proof to the following Theorem:

Theorem 2.25. *Fix any $k \geq 1$, $E \geq k$, and $r \in [1, (\log k)/2]$. There exists an r-round randomized protocol for* ExistsEqual *on vectors of length k that errs with probability $p_{\text{err}} = 2^{-(E+1)}$, using $O(Ek^{1/r})$ bits of communication.*

Proof. The number of tests per coordinate in phase j is l_j:

$$l_j = 2Ek^{j/r-1}.$$

Define $E_j = \sum_{j'=1}^{j} l_{j'}|I_{j'}|$ to be the portion of the error budget spent in phases 1 through j. We can express the asymptotic communication cost of the protocol in terms of the error budget as follows.

$$\begin{aligned} \sum_{j=1}^{r} l_j|I_{j-1}| &\leq l_1|I_0| + k^{1/r}\sum_{j=2}^{r} l_{j-1}|I_{j-1}| && l_j = k^{1/r}l_{j-1}. \\ &\leq 2Ek^{1/r} + E_{r-1}k^{1/r} && \text{Defn. of } E_{r-1}. \end{aligned}$$

Recall that the protocol terminates immediately after phase j if $E_j \geq E'$, which indicates $E_{r-1} < E'$. Hence, the total cost is bounded by

$$\leq (2E + E')k^{1/r} \;=\; O(Ek^{1/r}).$$

The protocol can only err if x and y differ in every coordinate. In this case, there are two possible sources of error. The first possibility is that the protocol answers *yes* because $|I_r| \geq 1$. By a union bound, this happens with probability at most

$$k2^{-\sum_{j=1}^{r} l_j} \leq k2^{-2E}.$$

The second possibility is that the protocol terminates prematurely and answers *yes* if $E_j \geq E'$ for some $j \in [1, r]$. The probability of this event occuring is also $\ll 2^{-E}$; see Eqn. (2.4). This concludes the proof. □

Proof of Theorem 2.16. Theorem 2.16 follows directly by combining Theorem 2.22 and Theorem 2.25. □

Remark 2.26. *By applying the reduction of Theorem 2.1 to Theorem 2.25, we conclude that* SetDisjointness *can be solved in $r+1$ rounds using $O(Ek^{1/r})$ bits of communication. In this particular case we actually do not need Theorem 2.1; it is possible*

to solve SetDisjointness *directly in r rounds with $O(Ek^{1/r})$ communication by an algorithm along the lines of Theorem 2.25 or [130]. Theorem 2.1 can also be applied to Theorem 2.24 to yield a* SetIntersection *protocol using $r+1$ rounds and $O(rEk^{1/r})$ communication, but here we do not see how to solve the problem directly in r rounds. It seems we would need some analogue of Lemma 2.20 tailored to the* SetIntersection *problem.*

2.4.4 A Communication Optimal EqualityTesting Protocol

Suppose we want a communication optimal EqualityTesting protocol using $O(k+E)$ bits. When $E \geq k$ we need $r = \Omega(\log k)$ rounds, by Theorem 2.4. In this section, we give a protocol for EqualityTesting that uses $O(r)$ rounds (rather than r) and $O(Ek^{1/r})$ bits of communication, assuming $E \geq k$. Observe that when $r = \Theta(\log k)$, there is no (asymptotic) difference between r rounds and $O(r)$ rounds as this only influences the leading constant in the communication volume.

2.4.4.1 Overview of the protocol

The protocol uses the concept of an *error budget* introduced in Section 2.4.3. To shave the factor r off the communication volume, we cannot afford to use $Ek^{j/r-1}$ test bits for each coordinate that participates in phase j. Consequently, we cannot guarantee with high probability (say $1-2^{-E}/r$) that the number of false positives is less than $k^{1-j/r}$.

Our protocol needs to be able to respond to the rare event that the number of false positives in I_j is larger than k_j. Notice that this type of error cannot be detected in the first j phases, and is not easily detectable in the following phases. The danger in the number of false positives in I_j exceeding k_j is that when the test bits for phase $j+1$ are exchanged using Lemma 2.20, the protocol may silently fail, with all test bits potentially corrupted.

To address these challenges, Alice and Bob each keep a *history* of all the test bits they have generated so far. They also keep a history of the test bits they have received from the other party, *which may have been corrupted*. Define T_A and T_B to be the true history of the test bits generated by Alice and Bob, respectively. Define $T_B^{(A)}$ to be what Alice believes Bob's history to be, and define $T_A^{(B)}$ analogously. Observe that if every invocation of Lemma 2.20 succeeds, then $T_A = T_A^{(B)}$ and $T_B = T_B^{(A)}$.

To detect inconsistencies, after Alice and Bob generate and exchange their test bits for phase j, they accumulate their views of the history into strings $T^{(A)} = T_A \circ T_B^{(A)}$

and $T^{(B)} = T_A^{(B)} \circ T_B$, respectively, where $\circ$ is the concatenation operator, and verify that $T^{(A)} = T^{(B)}$ with a certain number of inner product tests. This is called a *history check*. If the history check passes, they can proceed to phase $j+1$. If the history check fails then the results of phase j are junk, and we can infer that one of two types of low probability events occurred in phase $j-1$. The first possibility is that the test bits at phase $j-1$ were exchanged successfully (and consequently, the history check succeeded), but I_{j-1} contains more than k_{j-1} false positives. The second possibility is that Alice's and Bob's histories were already inconsistent at phase $j-1$, but the phase-$(j-1)$ history check failed to detect this. Notice that Alice and Bob cannot detect which of these types of errors occurred. In either case, we must undo the effects of phases j and $j-1$ and restart the protocol at the beginning of phase $j-1$. It may be that the history check then fails at the re-execution of phase $j-1$, in which case we would continue to rewind to the beginning of phase $j-2$, and so on. Being able to rewind multiple phases is important because we do not know which phase suffered the *first* error.

Both parties maintain an empirical *error meter* E'' that measures the sum of logarithms of probabilities of low probability (error) events that have been detected. If the error meter ever exceeds the error budget $E' = \Theta(E)$ we terminate the protocol, which we show occurs with probability $\ll 2^{-E}$. Thus, the process above (proceeding iteratively with phases, undoing and redoing them when errors are detected) must end by either successfully completing phase r or exceeding the error budget.

If Alice and Bob successfully finish phase r, we are still not done. This is because an error can happen in the later phases but we do not have sufficiently high $(1-2^{-E})$ confidence that they all succeeded. To build this confidence, Alice and Bob do inner product tests on the whole history, gradually increasing their number until $\Theta(E)$ tests have been done. If one of these history checks fails, we increase the error meter E'' appropriately and rewind the protocol to a suitable phase j in the first stage of the protocol.

Let us make every step of this protocol more quantitatively precise.

- The protocol has two stages, the *Refutation Stage* (in which potential equalities are refuted) and the *Verification Stage*, each consisting of a series of *phases*. Although the Refutation Stage logically precedes the Verification Stage, because phases can be undone, an execution of the protocol may oscillate between Refutation and Verification multiple times.

- The Refutation Stage is similar to the protocol in Section 2.4.2 except Alice and

Bob will verify whether the messages conveyed by Lemma 2.20 are successfully received with further inner product tests. The *budget* of phase j is

$$B_j = \frac{Ek_0^{1/r}}{\min\{j^2, r\}}.$$

Observe that $\sum_{j'=1}^{r} B_{j'} = O(Ek_0^{1/r})$. Thus, in phase j, we perform $l_j = B_j/k_{j-1}$ independent inner product tests on each coordinate in I_{j-1}. As usual, I_0 is initially $[k]$ and $k_j = k_0^{1-j/r}$, where $k_0 \leq E$. All histories $T_A, T_B, T_B^{(A)}, T_A^{(B)}$ are initially empty, and the error meter E'' is initially zero.

- Phase j has two steps, the *test step* and the *history check step*. In the test step, Alice and Bob conduct inner product tests as in Section 2.4.2, i.e.,, they generate l_j test bits for each coordinate in I_{j-1} and exchange them using Lemma 2.20, assuming their Hamming distance is at most k_{j-1}. Alice appends the test bits she generates onto the history T_A, and appends the test bits she receives from Bob onto $T_B^{(A)}$. Bob does likewise. In the history check step, they use B_j independent inner product tests to check whether $T^{(A)} = T^{(B)}$, where $T^{(A)} = T_A \circ T_B^{(A)}$, and $T^{(B)} = T_A^{(B)} \circ T_B$. The history check *fails* if they detect inequality and *passes* otherwise. Since B_j is, in general, less than E, we are still skeptical of history checks that pass.

- If the history check for phase j passes, Alice and Bob proceed to phase $j+1$, or proceed to the Verification Stage if $j = r$. Otherwise, an error has been detected: either the number of false positives in I_{j-1} is at least k_{j-1}, or the history check at phase $j-1$ *mistakenly* passed. The latter occurs with probability $\exp(-B_{j-1})$ and we show the former occurs with probability $\exp(-3k_0^{-1/r}B_{j-1}/4)$. Not knowing which occurred, we increment the error meter E'' by $k_0^{-1/r}B_{j-1}/2$ due to a union bound. If E'' exceeds the *error budget* $E' = cE$ then we halt, where $c \geq 2$ is a suitable constant. Otherwise we retract the effects of phases j and $j-1$ and continue the protocol at the beginning of phase $j-1$, with "fresh" random bits so as not to recreate previous errors.

- Observe that after phase r of the Refutation Stage, each coordinate in I_r has only passed about $B_r/k_{r-1} = E/r$ inner product tests, which is not high enough. Before the Verification Stage begins, Alice and Bob each generate E' test bits for each coordinate in I_r and append them to $T^{(A)}$ and $T^{(B)}$. (This can be viewed as a degenerate instantiation of Lemma 2.20 with $d = 0$, which requires

no communication.) If there are no false positives in I_r, these test bits must be identical.

- In the Verification Stage the phases are indexed in reverse order: $r, r-1, \ldots, 1$. In each successive phase j, Alice and Bob test the equality $T^{(A)} = T^{(B)}$ with B_j independent inner product tests. This process stops if it passes a total of E' tests, in which case they report that x and y are *equal* on I_r and *not equal* on $[k]\backslash I_r$, or some Verification phase j detects that $T^{(A)} \neq T^{(B)}$. In this case, we know Verification phases $r, r-1, \ldots, j+1$ passed *in error*, and that there must also have been an error in Refutation phase r. Therefore, Alice and Bob increment E'' by $k_0^{-1/r} B_r/2 + \sum_{j'=j+1}^{r} B_{j'}$ and halt if $E'' \geq E'$. If not, they rewind the execution of the protocol to phase j of the Refutation Stage and continue.

Algorithm 1 recapitulates this description in the form of pseudocode, from the perspective of Alice. Here $T_A[j, i]$ refers to the sequence of Alice's test bits in T_A for the ith coordinate produced in the most recent execution of phase j, and $T_A[j_1 \cdots j_2, \cdot]$ refers to the test bits generated from phase j_1 to phase j_2. Phase $r+1$ refers to the $E' \times |I_r|$ test bits generated between the Refutation and Verification stages. $T^{(A)}[j, i]$ refers to the concatenation of $T_A[j, i]$ and $T_B^{(A)}[j, i]$. What remains is to analyze the error probability of the protocol.

2.4.4.2 Analysis

To prove Theorem 2.17, it suffices to prove the following Theorem 2.27.

Theorem 2.27. *Fix any $k \geq 1$, $E \geq 1$, and $r \in [1, (\log k_0)/6]$, where $k_0 = \min\{k, E\}$. There exists a randomized protocol for* EqualityTesting *length-k vectors x, y with Hamming distance $\mathrm{dist}(x, y) \leq k_0$ that uses $O(r)$ rounds, $O(k + Ek_0^{1/r})$ bits of communication, and errs with probability $p_{\mathrm{err}} = 2^{-(E+1)}$.*

The protocol of Lemma 2.20 *fails* if Bob does not generate the correct $x' = x$, which indicates that the precondition is not met, i.e.,, $\mathrm{dist}(x, y) > d$. Refutation phase j *fails* if the condition in line 29 is not satisfied and the else branch at line 39 is executed in order to resume the protocol from phase $j-1$. Similarly, we say Verification phase j *fails* if the condition in line 47 is not satisfied, which also indicates the else branch at line 51 is executed and the protocol is resumed from Refutation phase j.

We begin the proof by showing that the extra communication caused by redoing some of the Refutation/Verification phases is properly covered by the total error budget. The following two lemmas actually prove that the error budget spent so far is correctly lower bounded in line 40 and line 52, and then Lemma 2.30 upper bounds the total number of extra phases by $O(r)$ and the overall extra communication by $O(k + Ek_0^{1/r})$.

Lemma 2.28. *Fix any $j \in [2, r]$. If phase j of the Refutation Stage fails, then the outcome of the most recent execution of phase $j-1$ happened with probability at most $\exp(-k_0^{-1/r} B_{j-1}/2)$.*

Proof. Recall that there are two types of errors at phase $j-1$. If the $(j-1)$th history check erroneously passed, this occurred with probability $\exp(-B_{j-1})$. The probability that more than k_{j-1} false positives survive in I_{j-1} is less than

$$\begin{aligned}
\binom{k_0}{k_{j-1}} 2^{-k_{j-1} l_{j-1}} &\le \left(\frac{ek_0}{k_{j-1}}\right)^{k_{j-1}} 2^{-k_{j-1} l_{j-1}} && \text{Because } \tbinom{n}{k} \le \left(\tfrac{en}{k}\right)^k. \\
&\le 2^{2k_{j-1}\log(k_0/k_{j-1}) - k_{j-1} l_{j-1}} && e \le k_0^{1/r} \le \tfrac{k_0}{k_{j-1}} \text{ due to } r \le \tfrac{\log k_0}{6}. \\
&\le 2^{-3 l_{j-1} k_{j-1}/4}, &&
\end{aligned}$$

Algorithm 1 An EqualityTesting protocol for Theorem 2.27 (from the perspective of Alice).

```
1:  procedure EqualityTesting                                  ▷ main procedure
2:      I_0 ← [k]
3:      k_0 ← min{k, E}                          ▷ initial bound on Hamming distance
4:      E' ← cE                                                ▷ error budget
5:      E'' ← 0                                                ▷ error meter
6:      for j ← 1, ..., r do
7:          B_j ← Ek_0^{1/r} / min{j^2, r}                     ▷ phase j communication budget
8:          k_j ← k_0^{1−j/r}                    ▷ ideal upper bound on Hamming distance
9:          l_j ← B_j / k_{j−1}                                ▷ tests per coordinate
10:     end for
11:     Refutation(1)
12:     Verification(r)
13:     for i ← 1, ..., k do
14:         output equal on coordinates I_r and not equal on [k]\I_r
15:     end for
16: end procedure
```

Algorithm 1 An EqualityTesting protocol for Theorem 2.27 (from the perspective of Alice).(cont.)

procedure InnerProductTest(w,b)
    perform b independent inner product tests on w and return the test bits
end procedure

procedure Refutation(j) ▷ phase j of the Refutation Stage
    $T_A[j,\cdot] \leftarrow \perp$
    for all $i \in I_{j-1}$ **do**
        $T_A[j,i] \leftarrow$ InnerProductTest(x_i, l_j)
    end for
    send $T_A[j,\cdot]$ to Bob and receive $T_B^{(A)}[j,\cdot]$ from Bob via Corollary 2.21
    $T^{(A)}[j,\cdot] \leftarrow T_A[j,\cdot] \circ T_B^{(A)}[j,\cdot]$
    $\hat{T}^{(A)} \leftarrow$ InnerProductTest$(T^{(A)}[1 \cdots j,\cdot], B_j)$
    send $\hat{T}^{(A)}$ to Bob and receive $\hat{T}^{(B)}$ from Bob directly
    if $\hat{T}^{(A)} = \hat{T}^{(B)}$ **then**
        $I_j \leftarrow \{i \in I_{j-1} \mid T_A[j,i] = T_B^{(A)}[j,i]\}$
        if $j < r$ **then**
            Refutation$(j+1)$
        else
            $T^{(A)}[r+1,\cdot] \leftarrow \perp$
            for all $i \in I_r$ **do**
                $T^{(A)}[r+1,i] \leftarrow$ InnerProductTest(x_i, E')
            end for
        end if
    else
        $E'' \leftarrow E'' + k_0^{-1/r} B_{j-1}/2$, and terminate if $E'' \geq E'$
        Refutation$(j-1)$
    end if
end procedure

procedure Verification(j) ▷ phase j of the Verification Stage
    $\hat{T}^{(A)} \leftarrow$ InnerProductTest$(T^{(A)}[\cdot,\cdot], B_j)$
    send $\hat{T}^{(A)}$ to Bob and receive $\hat{T}^{(B)}$ from Bob directly
    if $\hat{T}^{(A)} = \hat{T}^{(B)}$ **then**
        if $\sum_{j'=j}^{r} B_{j'} < E'$ **then**
            Verification$(j-1)$
        end if
    else
        $E'' \leftarrow E'' + k_0^{-1/r} B_r/2 + \sum_{j'=j+1}^{r} B_{j'}$, and terminate if $E'' \geq E'$
        Refutation(j)
        Verification(r)
    end if
end procedure

where the last step follows from the inequality

$$\begin{aligned}
2\log\frac{k_0}{k_{j-1}} &= 2(j-1)\log k_0^{1/r} \\
&\le \frac{8^{j-1}\log k_0^{1/r}}{4(j-1)^2} && \text{Because } 8x^3 \le 8^x \text{ for } x \in \mathbb{N}. \\
&\le \frac{k_0^{(j-1)/r}}{4(j-1)^2} && \log k_0^{1/r} \le \tfrac{k_0^{1/r}}{8} \text{ due to } k_0^{1/r} \ge 2^6. \\
&\le \frac{B_{j-1}}{4k_{j-2}} && \text{Defn. of } B_{j-1}. \\
&= \frac{l_{j-1}}{4}. && \text{Defn. of } l_{j-1}.
\end{aligned}$$

Combining the above two cases, by a union bound, the outcome of the most recent execution of phase $j-1$ of the Refutation Stage happens with probability at most $\exp(-B_{j-1}) + \exp(-3l_{j-1}k_{j-1}/4) = \exp(-B_{j-1}) + \exp(-3k_0^{-1/r}B_{j-1}/4) \le \exp(-k_0^{-1/r}B_{j-1}/2)$, as claimed. □

Lemma 2.29. *Fix any $j \in [1, r]$. If phase j of the Verification Stage fails, then the outcomes of the most recent execution of phases $r, r-1, \ldots, j+1$ of the Verification Stage and phase r of the Refutation Stage happened with overall probability at most $\exp(-k_0^{-1/r}B_r/2 - \sum_{j'=j+1}^{r} B_{j'})$.*

Proof. Notice that the failure of Verification phase j means all previous Verification phases $r, r-1, \ldots, j+1$ failed to detect an inconsistency in the history, which occurs with probability $\exp(-\sum_{j'=j+1}^{r} B_{j'})$. Meanwhile, the inconsistency is caused by an error of some type in Refutation phase r, which, according to Lemma 2.28, occurs with probability at most $\exp(-k_0^{-1/r}B_r/2)$. Therefore, the outcomes of the most recent execution of Verification phases $r, r-1, \ldots, j+1$ and Refutation phase r happened with overall probability at most $\exp(-k_0^{-1/r}B_r/2 - \sum_{j'=j+1}^{r} B_{j'})$. □

Lemma 2.30. *Algorithm 1 executes $O(r)$ extra Refutation/Verification phases and uses $O(k + Ek_0^{1/r})$ extra bits of communication.*

Proof. We first consider the total number of *extra* phases. Each failure of Refutation phase j uses at least $k_0^{-1/r}B_{j-1}/2 \ge E/(2r)$ of the error budget and causes the re-execution of two phases, namely $j-1$ and j. Similarly, each failure of Verification phase j uses $k_0^{-1/r}B_r/2 + \sum_{j'=j+1}^{r} B_{j'} \ge (r-j+1)E/(2r)$ of the error budget and causes the re-execution of $2(r-j+1)$ phases. Thus, the total number of extra phases is at most $4cr = O(r)$, where the error budget $E' = cE$.

Turning to the overall *extra* communication, notice that phase j of the Refutation Stage has communication volume $O(B_j + k_{j-1}\log(k/k_{j-1}))$ and phase j of the Verification Stage has communication volume $O(B_j)$. For any $j \in [2, r]$, also notice that $B_{j-1}/B_j \leq j^2/(j-1)^2 \leq 4 \leq k_0^{1/r}$. Thus, the communication caused by each failure is at most $O(k_0^{1/r})$ times the error budget spent by that failure, if we temporarily ignore the $k_{j-1}\log(k/k_{j-1})$ term.

In order to upper bound the communication contributed by the $k_{j-1}\log(k/k_{j-1})$ term, observe that Refutation phase j can only be repeated $O(j^2)$ times before the error budget is exhausted. Thus, the overall extra communication is upper bounded by $O(k + Ek_0^{1/r})$ since

$$
\begin{aligned}
&O(k_0^{1/r}) \cdot E' + \sum_{j=1}^{r} O(j^2) \cdot k_{j-1} \log \frac{k}{k_{j-1}} \\
&= O(k_0^{1/r}) \cdot E' + k_0 \sum_{j=1}^{r} \frac{O(j^2)}{k_0^{(j-1)/r}} \left(\log \frac{k}{k_0} + \log k_0^{(j-1)/r} \right) \\
&= O(k_0^{1/r}) \cdot E' + k_0 \log \frac{k}{k_0} \sum_{j=1}^{r} \frac{O(j^2)}{k_0^{(j-1)/r}} + k_0 \sum_{j=1}^{r} \frac{O(j^2) \cdot \log k_0^{(j-1)/r}}{k_0^{(j-1)/r}} \\
&= O(k + Ek_0^{1/r}). && \text{(Because } k_0^{1/r} \geq 2^6 \text{ and } k_0 \leq k.\text{)}
\end{aligned}
$$

□

Now we are ready to prove Theorem 2.27.

Proof of Theorem 2.27. If there are no errors, Algorithm 1 has at most $2r$ phases and uses $O(\sum_{j=1}^{r}(B_j + k_{j-1}\log(k/k_{j-1}))) = O(k + Ek_0^{1/r})$ communication, where each phase can be implemented in $O(1)$ rounds. Together with Lemma 2.30, we have shown that it is an $O(r)$-round randomized EqualityTesting protocol using $O(k + Ek_0^{1/r})$ bits of communication. Thus, it suffices to calculate the error probability of the protocol.

Consider a possible execution of the protocol, i.e.,, the sequence of the Refutation/Verification phases that are performed. It can be represented by a *unique* 0-1 string of length at most $4cr + 2r$ (by the proof of Lemma 2.30) such that each "1" corresponds to a failed phase. In particular, each execution of the protocol that terminates prematurely because $E'' \geq E'$ is represented as a 0-1 string, which occurs with probability at most $2^{-E'}$, by Lemmas 2.28 and 2.29. Hence the overall probability of terminating prematurely is $2^{4cr+2r} \cdot 2^{-E'}$.

An error can also be caused by at least one false positive surviving all E' independent inner product tests generated after Refutation phase r. The probability of this

happening is at most $k_0 2^{-E'}$. The last possible source of error is that all Verification phases fail to detect the inequality $T^{(A)} \neq T^{(B)}$. According to line 48, the probability of this happening is at most $2^{-E'}$. Hence, the overall probability of error is upper bounded by

$$2^{4cr+2r} \cdot 2^{-E'} + k_0 2^{-E'} + 2^{-E'} = \text{poly}(k_0) 2^{-E'},$$

which is at most 2^{-E} for, say, $E' = 2E$. This concludes the proof. □

Proof of Theorem 2.17. Theorem 2.17 subsequently follows by applying Theorem 2.22 and Theorem 2.27 in sequence. □

2.5 An $O(k + Ek^{1/r} \log r + Er \log r)$ bits protocol

As demonstrated in the previous two sections, there are several difficulties in using the notion of error budget from our ExistsEqual equal protocol to shave off the factor r in our $O(rEk^{1/r} + k)$ bits protocol for EqualityTesting. First, since we cannot distinguish between equal coordinates from false positives, it is difficult to measure the fraction of the error budget spent in a particular round as soon as the round finishes. Recall that in ExistsEqual, we can assume every coordinate in I_j are false positives by making the default answer of the protocol "yes". Second, the test bit exchange protocol in Corollary 2.21 requires an apriori upper bound d on the Hamming distance of the test bits, i.e., the number of false positives discovered in the current round. It can fail silently when the actual number of discovered false positives is larger than d. Detecting failed test bit exchange protocols with high enough confidence, i.e., $1 - 2^{-E}$ can be difficult given our communication budget.

In this section, we present another approach to address these two problems. To address the first problem, for each communication round j starting from the second round, we carry out $\log r$ parallel and independent tests with different "k_{j-1}'s" and "l_j's" to accommodate different number of false positives left in I_{j-1}. More specifically, the first parallel test uses k_{j-1} as the Hamming distance bound and l_j as the number of test bits per element. For simplicity, we refer to these two parameters as the *width* and the *height* of a test. The i-th parallel test will use a width $k_{j-1}^{(i)} = k_{j-1}/2^{i-1}$ and height $l_j^{(i)} = 2^{i-1} l_j$. The "area" of each test is always $k_{j-1} l_j$. The main purpose of these parallel tests is to measure the amount of error budget we spent on the *previous* round. In particular, if the actual number of false positives discovered is small so the adversary has only spent little error budget in the previous round, the parallel tests with smaller width and larger height will succeed and we can obtain more test bits per

element for I_{j-1}, which in turn decreases the maximum possible width to compensate for the bigger leftover error budget. On the other hand, a larger Hamming distance will imply a larger error budget spent in the previous round and less error budget remains, which also get us closer to the finish line.

To address the second problem, with each parallel test, we also transmit a $\theta(E)$ bits hash on the test bit we are to exchange. After receiver receives the test bits using Corollary 2.21, he will hash the test bits using the same randomness and verify it against the hash sent by the sender. Hence, with confidence at least $1 - 2^{-\Omega(E)}$, if the exchange protocol fails, this test will be able to detect the event.

Now let us state the protocol more formally. Similar to before, we first invoke Theorem 2.22 to reduce the number of false positive to be at most E using $O(k)$ communication and $\log^*(k/E)$ rounds. Hence, $k_0 \leq \min\{k, E\}$ is an initial upper bound on the number of false positives. Then we state the main theorem:

Theorem 2.31. *Fix any $k \geq 1$, $E \geq 1$, and $r \in [1, (\log k_0)/2]$, where $k_0 \leq \min\{k, E\}$. There exists a randomized protocol for* EqualityTesting *length-k vectors x, y with Hamming distance $\mathrm{dist}(x, y) \leq k_0$ that uses r rounds, $O(k + Ek_0^{1/r} \log r + Er \log r)$ bits of communication, and errs with probability $p_{\mathrm{err}} = 2^{-(E+1)}$.*

Now let E' denote the initial total error budget and we choose $E' = 7E$ here. For the first round, we only conduct one single parallel test with width k_0 and height $l_1 = 7Ek_0^{1/r-1}$. Since the overall error budget is $7E$, the maximum number of false positives allowed at the end of the first round is $E'/l_1 = k_0^{1-1/r}$. Suppose the *actual* number of false positives at the end of the round j is $k_j^* = \beta_j' k_j$ for some $0 \leq \beta_j' \leq 1$. Then at round $j+1$, the first $i^* = \max\{\lfloor \log \frac{1}{\beta_j'} \rfloor + 1, \log r\}$ parallel tests must succeed because they all have valid widths, and the (i^*+1)-th parallel test might fail since their width is too small[8]. Let $\beta_j = \max\{\beta_j', 1/(2r)\}$.Hence, we can obtain at least $l_{j+1}^{(i^*)} \geq l_{j+1}/(2\beta_j)$ test bits per element in I_j in round $j+1$. For simplicity, we set $l_{j+1}^* = l_{j+1}/(2\beta_j)$ and $l_1^* = l_1$. This in turns allow us to calculate the maximum number of allowed positive for round j by setting $k_j = E'/l_j^*$.

Now we let $k_{j-1} l_j = 14\alpha_j E k_0^{1/r}$ for some constant α_j. The protocol can be seen as a scheme where in round $j \geq 2$, the receiver learns (an upper bound to) β_{j-1} and uses the information to decide the communication budget the receiver should spend next round, i.e., α_{j+1}. We set $\alpha_1 = 1$ and $\alpha_2 = 1/2$. Suppose we wish the total communication budget to be $14CEk^{1/r} \log r$ for some constant C, then we wish to set

[8]These tests can still succeed because some number of false positives last round can still remain false positives this round and not contributing to the Hamming distance.

α_j in a way that $\sum_{j=1}^{r} \alpha_j \leq C$ and moreover, the number of test bits per element in I_{r-1} is at least $\Omega(E)$ so no false positives survive with high probability.

This leads to the following equalities:

$$\begin{aligned}
k_j &= \prod_{i=1}^{j-1} \frac{\beta_i}{\alpha_{i+1}} k_0^{1-j/r}. \\
k_j^* &= \beta'_j k_j \\
&= \beta'_j \prod_{i=1}^{j-1} \frac{\beta_i}{\alpha_{i+1}} k_0^{1-j/r}. \\
l_j &= 14\alpha_j E k_0^{1/r} / k_{j-1} \\
&= 14\alpha_j \prod_{i=1}^{j-2} \frac{\alpha_{i+1}}{\beta_i} E k^{j/r-1}. \\
l_j^* &= l_j/(2\beta_{j-1}) \\
&= 7 \prod_{i=1}^{j-1} \frac{\alpha_{i+1}}{\beta_i} E k^{j/r-1}.
\end{aligned}$$

Recall that our goal for setting α_j is to maximize the test bits per elements in I_{r-1} while obeying the budget constraint $\sum_{j=1}^{r} \alpha_j \leq C$. By AM-GM inequality, setting $\alpha_j = C/r$ can achieve the maximum. On the other hand, the goal of the adversary, who is setting β_j, is to make the number of test bits per element in the last round as small as possible, i.e., to maximize $\prod_{j=1}^{j} \beta_j$. Again, his best strategy is to let all β_j be equal. Since we have $\sum_j \beta'_j \leq 1$ and $\sum_j \beta_j \leq \sum_j (\beta'_j + 1/r) \leq 2$. By setting $C = 2$ and $\alpha_j = \beta_{j-2}$, we can achieve the goal of making the height of the final test at least $7E$. Under this scheme, we have:

$$k_j = \begin{cases} k^{1-1/r} & \text{when } j = 1. \\ \beta_{j-1} k^{1-j/r} & \text{otherwise.} \end{cases}$$

$$k_j^* = \begin{cases} \beta_1' k^{1-1/r} & \text{when } j = 1. \\ \beta_j' \beta_{j-1} k^{1-j/r} & \text{otherwise.} \end{cases}$$

$$l_j = \begin{cases} 7Ek^{1/r-1} & \text{when } j = 1. \\ 14Ek^{j/r-1} & \text{otherwise.} \end{cases}$$

$$l_j^* = \begin{cases} 7Ek^{1/r-1} & \text{when } j = 1. \\ 14/\beta_{j-1} Ek^{j/r-1} & \text{otherwise.} \end{cases}$$

Now we show the total communication volume of the entire protocol is $O(Ek^{1/r} \log r + k + Er \log r)$. The communication is divided into three parts: 1. The $O(k)$ communication introduced by applying the protocol in Theorem 2.22 to reduce the number of false positive to be at most E. 2. The $O(Ek_0^{1/r} \log r + k)$ communication introduced by using Corollary 2.21 to exchange test bits across all r rounds. 3. The $O(Er \log r)$ bits check bitss used to verify the result of the test bits exchange protocol. The first part is proved in Theorem 2.22 and the third part follows naturally from construction. Now we prove the second part.

Lemma 2.32. *The total communication volume of the protocol introduced by exchanging all the test bits using Corollary 2.21 is* $O(Ek^{1/r} \log r + k)$.

Proof. Recall that Corollary 2.21 stated that the cost of exchanging two strings of block size l, length k and Hamming distance at most d is at most $O(dl + d \log(k/d))$. As previously, we will bound two terms separately. First we bound the first term:

Since $k_0 l_1 = k \cdot 7Ek^{1/r-1} = 7Ek^{1/r}$, we start to bound the sum from the second round:

$$\begin{aligned}
&\sum_{j=2}^{r}\sum_{i=1}^{\log r} k_{j-1}^{(i)} l_j^{(i)} \\
&= \sum_{j=2}^{r}\sum_{i=1}^{\log r} k_{j-1}/2^{i-1} \cdot (2^{i-1} l_j) \\
&= \sum_{j=2}^{r} \log r \cdot k_{j-1} l_j \\
&= \log r \cdot k^{1-1/r} \cdot (14Ek_0^{2/r-1}) + \log r \sum_{j=3}^{r} \beta_{j-2} k_0^{1-(j-1)/r} 14Ek_0^{j/r-1} \\
&\qquad\qquad \text{(Extracting the first term.)} \\
&= 14Ek_0^{1/r} \log r + 14Ek_0^{1/r} \log r \sum_{j=1}^{r-2} \beta_j \qquad \text{(Replacing } j-2 \text{ with } j.\text{)} \\
&\le 14Ek_0^{1/r} \log r + 14Ek_0^{1/r} \log r \sum_{j=1}^{r-2} (\beta'_j + 1/r) \qquad (\beta_j \le \beta'_j + 1/r.) \\
&\le 14Ek_0^{1/r} \log r + 28Ek_0^{1/r} \log r \qquad (\textstyle\sum_{j=1}^{r} \beta'_j \le 1) \\
&= 42Ek_0^{1/r} \log r.
\end{aligned}$$

Next we bound the second term. Since for any $j \ge 2$, we have

$$\begin{aligned}
&\sum_{i=1}^{\log r} k_{j-1}^{(i)} \log(k/k_{j-1}^{(i)}) \\
&= \sum_{i=1}^{\log r} \frac{k_{j-1}}{2^{i-1}} \log \frac{2^{i-1}k}{k_{j-1}} \\
&= k_{j-1} \log \frac{k}{k_{j-1}} \sum_{i=1}^{\log r} \frac{1}{2^{i-1}} + k_{j-1} \sum_{i=1}^{\log r} \frac{i-1}{2^{i-1}} \\
&= O(k_{j-1} \log \frac{k}{k_{j-1}}).
\end{aligned}$$

It suffices to only consider the first parallel test from each round. Now we bound the overall communication across all rounds. For the first two rounds, we have: $k_0 \log k/k_0 < k$ and $k_1 \log k/k_1 = k^{1-1/r} \log k/k^{1-1/r} < k$ for any $k^{1/r} \ge 2$. So we start the bound from $j = 3$.

$$
\begin{aligned}
&\sum_{j=3}^{r} k_{j-1} \log \frac{k}{k_{j-1}} \\
&= \sum_{j=2}^{r-1} \beta_{j-1} k_0^{1-j/r} \log \frac{k}{\beta_{j-1} k_0^{1-j/r}} \\
&\leq \sum_{j=2}^{r-1} \beta_{j-1} k \frac{\log \frac{k^{j/r}}{\beta_{j-1}}}{k^{j/r}} \\
&\leq k \sum_{j=2}^{r-1} \frac{\log k^{j/r}}{k^{j/r}} + k \sum_{j=2}^{r-1} \frac{\beta_{j-1} \log \frac{1}{\beta_{j-1}}}{k^{j/r}} && (\beta_j \leq 1.) \\
&\leq k \sum_{j=2}^{r-1} \frac{\log k^{j/r}}{k^{j/r}} + \frac{k}{e} \sum_{j=2}^{r-1} \frac{1}{k^{j/r}} \\
&= O(k). && (k^{1/r} \geq 2.)
\end{aligned}
$$

This shows that the total communication volume introduced by applying Corollary 2.21 is $O(Ek_0^{1/r} \log r + k) = O(Ek^{1/r} \log r + k)$, finishing the proof. □

Now we show that the error probability of our protocol is less than 2^{-E}.

Lemma 2.33. *The protocol errs with probability at most 2^{-E-1}.*

Proof. First we bound the probability of any test-bit exchange protocol fails unnoticed. A failed parallel test passes the check bits test with probability at most 2^{-3E}. Taking union bound, the probability of any check bits test fails is at most $2^{-3E} r \log r \leq 2^{-E}/6$ for $r \leq \log k/2 \leq \log E$.

Now we bound the probability that we exceed the error budget at any particular stage of the algorithm. Similar to the ExistsEqual protocol, we want to argue that "configurations" of test bit that contains $> 3E$ false positives happen with probability at most $2^{-E}/3$. Here by "configuration" we mean the outcome of all test bits for a particular unequal coordinate until the first unequal test bit is observed. The equal test bits are considered false positives. Each configuration can be uniquely described by the number of false positive we see for each coordinate. Since each configuration happens with probability at most 2^{-3E}, the ovarall probability, by union bound, is at most:

$$
\binom{E' + k - 1}{k - 1} 2^{-E'} \leq \exp(k \log (e(E' + k)/k) - E') < 2^{-E}/6.
$$

For $E' \geq 7E$.

Finally we bound the probability that any unequal coordinate goes through the entire protocol unnoticed. Notice that in the final round, any coordinate in I_{r-1} goes through at least $l_r = 6E$ test bits. The probability that any of them passing all $6E$ test bits is at most $k_0 2^{-6E} \leq 2^{-E}/6$ for $k_0 \leq E$. Then by union bound, the probability that the protocol errs is at most 2^{-E-1}. □

This finishes the proof to Theorem 2.31, hence finishing the proof to Theorem 2.18.

2.6 Distributed Triangle Enumeration

One way to solve local triangle enumeration in the CONGEST model is to execute, in parallel, a SetIntersection protocol across every edge of the graph, where the set associated with a vertex is a list of its neighbors. Since there are at most $\Delta n/2$ edges, we need the SetIntersection error probability to be 2^{-E}, $E = \Theta(\log n)$, in order to guarantee a global success probability of $1 - 1/\operatorname{poly}(n)$. Our lower bound says any algorithm taking this approach must take $\Omega((\Delta + E\Delta^{1/r})/\log n + r)$ rounds since each round of CONGEST allows for one $O(\log n)$-bit message. The hardest situation seems to be when $\Delta = E = \Theta(\log n)$, in which case the optimum choice is to set $r = \log \Delta$, making the triangle enumeration algorithm run in $O(\log \Delta) = O(\log \log n)$ time. In Theorem 2.34 we show that it is possible to handle this situation exponentially faster, in $O(\log \log \Delta) = O(\log \log \log n)$ time, and in general, to solve local triangle enumeration [89] in optimal $O(\Delta/\log n)$ time so long as $\Delta > \log n \log \log \log n$.

Theorem 2.34. *Local triangle enumeration can be solved in a CONGEST network $G = (V, E)$ with maximum degree Δ in $O(\Delta/\log n + \log \log \Delta)$ rounds with probability $1 - 1/\operatorname{poly}(n)$. This is optimal for all $\Delta = \Omega(\log n \log \log \log n)$.*

Proof. The algorithm consists of $\min\{\log \log \Delta, \log \log \log n\}$ phases. The goal of the first phase is to transform the original triangle enumeration problem into one with maximum degree $\Delta_1 < (\log n)^{o(1)}$, in $O(\log^* n)$ rounds of communication. The goal of every subsequent phase is to reduce the maximum degree from $\Delta' \leq \sqrt{\log n}$ to $\sqrt{\Delta'}$, in $O(1)$ rounds of communication. Thus, the total number of rounds is $O(\log \log \Delta)$ rounds if the first round is skipped, and $O(\log^* n + \log \log(\Delta_1)) = O(\log \log \log n)$ otherwise.

Phase One. Suppose $\Delta \geq \sqrt{\log n}$. Each vertex u is identified with the set $A_u = \{\mathsf{ID}(v) \mid \{v, u\} \in E\}$ having size Δ. For each $\{u, v\} \in E$ we reduce SetIntersection

to EqualityTesting by applying Theorem 2.1, then run the two-party EqualityTesting protocol of Theorem 2.15, with $k = \max\{\Delta, \log n\}, r = \log^* n$, and $E = r^{-1}k^{1-1/r}$. (I.e., if $\Delta < \log n$ we imagine padding each set to size $\log n$ with dummy elements.) One undesirable property of this protocol is that it can fail "silently" if the preconditions of Lemma 2.20 are not met. When the Hamming distance between two strings exceeds the threshold d, Bob generates a garbage string $x' \neq x$ but fails to detect this. To rectify this problem, we change the Lemma 2.20 protocol slightly: Alice sends the color $\phi(x)$ of her string, as well as an $O(\log n)$-bit hash $h(x)$. Bob reconstructs x' as usual and terminates the protocol if $h(x) \neq h(x')$. Clearly the probability of an undetected failure (i.e., $x \neq x'$ but $h(x) = h(x')$) is $1/\operatorname{poly}(n)$. Define $G_1 = (V, E_1)$ such that $\{u, v\} \in E_1$ iff the SetIntersection protocol over $\{u, v\}$ detected a failure. In other words, with high probability, all triangles in G have been discovered, except for those contained entirely inside G_1. The probability that any particular edge appears in E_1 is $2^{-E} = 2^{-k^{1-1/\log^* n}/\log^* n}$ and independent of all other edges. In particular, if $\Delta \gg (\log n)^{1+1/\log^* n}$ then no errors occur, with probability $1 - 1/\operatorname{poly}(n)$. Define Δ_1 to be the maximum degree in G_1. Thus,

$$\begin{aligned}
\Pr\left[\Delta_1 \geq (\log n)^{2\epsilon}\right] &\leq n \cdot \binom{\Delta}{(\log n)^{2\epsilon}} \cdot \left(2^{-E}\right)^{(\log n)^{2\epsilon}} \qquad (\epsilon = 1/r = 1/\log^* n) \\
&\leq n \cdot \exp(O((\log n)^{2\epsilon} \log\log n)) \cdot 2^{-\epsilon(\log n)^{1-\epsilon} \cdot (\log n)^{2\epsilon}} \\
&\leq 1/\operatorname{poly}(n).
\end{aligned}$$

Phases Two and Above. Suppose that at some round, we have detected all triangles except for those contained in some subgraph $G' = (V, E')$ having maximum degree $\Delta' < \sqrt{\log n}$. Express Δ' as $(\log n)^\gamma$, where $\gamma < 1/2$. We execute the EqualityTesting protocol of Theorem 2.24 with $k = \Delta'$, $r = 2$, and $E = C(\log n)^{1-\gamma/2}$ for a sufficiently large constant C. Note that $1 - \gamma/2 > \gamma$, so $E > k$, as required by Theorem 2.24. The protocol takes $O(Ek^{1/2}/\log n + r) = O(1)$ rounds since the communication volume is $O(Ek^{1/2}) = O(\log n)$ and $r = 2$. Let G'' be the subgraph of G' consisting of edges whose protocols detected a failure and Δ'' be the maximum degree in G''. Once again,

$$\begin{aligned}
\Pr\left[\Delta'' \geq (\log n)^{\gamma/2}\right] &\leq n \cdot \binom{\Delta'}{(\log n)^{\gamma/2}} \cdot \left(2^{-E}\right)^{(\log n)^{\gamma/2}} \\
&\leq n \cdot \exp(O((\log n)^{\gamma/2} \log\log n)) \cdot 2^{-C(\log n)^{1-\gamma/2} \cdot (\log n)^{\gamma/2}} \\
&\leq 1/\operatorname{poly}(n).
\end{aligned}$$

Thus, once $\Delta \leq \sqrt{\log n}$, $\log\log\Delta \leq \log\log\log n - 1$ of these 2-round phases suffice to find all remaining triangles in G. □

Theorem 2.34 depends critically on the duality between edges and SetIntersection instances, and between edge endpoints and elements of sets. In particular, when an execution of a SetIntersection over $\{u, v\}$ is successful, this effectively *removes* $\{u, v\}$ from the graph, thereby removing many occurrences of $\mathsf{ID}(u)$ and $\mathsf{ID}(v)$ from adjacent sets.

Consider a slightly more general situation where we have a graph of *arboricity* λ (but unbounded Δ), witnessed by a given acyclic orientation having out-degree at most λ. Redefine the set A_u to be the set of out-neighbors of u.

$$A_u = \{\mathsf{ID}(v) \mid \{u, v\} \in E \text{ with orientation } u \to v\}.$$

By definition $|A_u| \leq \lambda$. Because the orientation is acyclic, every triangle on $\{x, y, z\}$ is (up to renaming) oriented as $x \to y,\ x \to z,\ y \to z$. Thus, it will *only* be detectable by the SetIntersection instance associated with $\{x, y\}$.

Theorem 2.35. *Let $G = (V, E)$ be a CONGEST network equipped with an acyclic orientation with outdegree at most λ. We can solve local triangle enumeration on G in $O(\lambda/\log n + \log\lambda)$ time.*

Proof. We apply Theorem 2.1 to reduce each SetIntersection instance to an EqualityTesting instance, then apply Theorem 2.17 with $E = \Theta(\log n)$ and $r = \log\lambda$ to solve each with $O(\lambda + E\lambda^{1/r}) = O(\lambda + E)$ communication in $O((\lambda + E)/\log n + r) = O(\lambda/\log n + \log\lambda)$ time. Note that the dependence on λ here is exponentially worse than the dependence on Δ in Theorem 2.34. □

It may be that G is known to have arboricity λ, but an acyclic orientation is unavailable. The well known "peeling algorithm" (see [37] or [19]) computes a $C\lambda$ orientation in $O(\log_C n)$ time for C sufficiently large, say $C \geq 3$. Using this algorithm as a preprocessing step, we can solve local triangle enumeration optimally when $\lambda = \Omega(\log^2 n)$.

Theorem 2.36. *Let $G = (V, E)$ be a CONGEST network having arboricity λ (with no upper bound on Δ). Local triangle enumeration can be solved in optimal $O(\lambda/\log n)$ time when $\lambda = \Omega(\log^2 n)$, and sublogarithmic time $O(\log n/\log(\log^2 n/\lambda))$ otherwise.*

Proof. The algorithm computes a $\gamma \cdot \lambda$ orientation in $O(\log_\gamma n)$ time and then applies Theorem 2.35 to solve local triangle enumeration in $O(\gamma\lambda/\log n + \log(\gamma\lambda))$ time. The

only question is how to set γ. If $\lambda = \Omega(\log^2 n)$ we set $\gamma = 3$, making the total time $O(\lambda/\log n)$, which is optimal [89]. Otherwise we choose γ to balance the $\log_\gamma n$ and $\gamma\lambda/\log n$ terms, so that

$$\gamma \log \gamma = \log^2 n/\lambda$$

Thus, the total running time is slightly sublogarithmic $O(\log n/\log(\log^2 n/\lambda))$. Specifically, it is $O(\log n/\log\log n)$ whenever $\lambda < \log^{2-\epsilon} n$. □

2.7 Reductions and Near Equivalences

Brody et al. [23] proved that SetIntersection on sets of size k is reducible to EqualityTesting on vectors of length $O(k)$, at the cost of one round and $O(k)$ bits of communication. However, the reduction is *randomized* and fails with probability at least $\exp(-\tilde{O}(\sqrt{k}))$. This is the probability that when k balls are thrown uniformly at random into k bins, some bin contains $\omega(\sqrt{k})$ balls.

Recall the statement of Theorem 2.1:

$$\mathsf{Eq}(k, r, p_{\text{err}}) \le \mathsf{SetInt}(k, r, p_{\text{err}}), \qquad \mathsf{SetInt}(k, r+1, p_{\text{err}}) \le \mathsf{Eq}(k, r, p_{\text{err}}) + \zeta,$$
$$\exists\mathsf{Eq}(k, r, p_{\text{err}}) \le \mathsf{SetDisj}(k, r, p_{\text{err}}), \qquad \mathsf{SetDisj}(k, r+1, p_{\text{err}}) \le \exists\mathsf{Eq}(k, r, p_{\text{err}}) + \zeta,$$

where $\zeta = O(k + \log\log p_{\text{err}}^{-1})$. In other words, under any error regime p_{err}, the communication complexity of SetIntersection and EqualityTesting are the same, up to one round and $O(k + \log\log p_{\text{err}}^{-1})$ bits of communication, and that the same relationship holds between SetDisjointness and ExistsEqual. The proof is inspired by the probabilistic reduction of Brody et al. [23], but uses succinct encodings of perfect hash functions rather than random hash functions.

Proof of Theorem 2.1. The leftmost inequalities have been observed before [130, 23]. Given inputs x, y to ExistsEqual or EqualityTesting, Alice and Bob generate sets $A = \{(1, x_1), \ldots, (k, x_k)\}$ and $B = \{(1, y_1), \ldots, (k, y_k)\}$ before the first round of communication and then proceed to solve SetIntersection or SetDisjointness on (A, B). Knowing $A \cap B$ or whether $A \cap B = \emptyset$ clearly allows them to determine the correct output of EqualityTesting or ExistsEqual on (x, y).

The reverse direction is slightly more complicated. Let (A, B) be the instance of SetIntersection or SetDisjointness over a universe U with size at most $|U| = O(k^2/p_{\text{err}})$. Alice examines her set A, and picks a *perfect* hash function $h : U \mapsto [k]$ for A, i.e., h is injective on A. (This can be done in $O(k)$ time, in expectation, using only

private randomness. In principle Alice could do this step deterministically, given sufficient time.) Most importantly, h can be described using $O(k + \log\log|U|) = O(k + \log\log p_{\text{err}}^{-1})$ bits [131], using a variant of the Fredman-Komlós-Szemerédi [66] 2-level perfect hashing scheme.[9] Alice sends the $O(k + \log\log p_{\text{err}}^{-1})$-bit description of h to Bob. Bob calculates $B_j = B \cap h^{-1}(j)$ and responds to Alice with the distribution $|B_0|, |B_1|, \ldots, |B_{k-1}|$, which takes at most $2k$ bits. They can now generate an instance of Equality Testing where the k equality tests are the pairs $A_0 \times B_0, A_1 \times B_1, \ldots, A_{k-1} \times B_{k-1}$. By construction, $A_j = A \cap h^{-1}(j)$ is a 1-element set. There is clearly a 1-1 correspondence between equal pairs and elements in $A \cap B$. We have Bob speak first in the EqualityTesting/ExistsEqual protocol; thus, the overhead for this reduction is just 1 round of communication and $O(k + \log\log p_{\text{err}}^{-1})$ bits. □

2.8 Conclusions and Open Problems

We have established a new three-way tradeoff between rounds, communication, and error probability for many fundamental problems in communication complexity such as SetDisjointness and EqualityTesting. Our lower bound is largely incomparable to the round-communication lower bounds of [130, 23], and stylistically very different from both [130] and [23]. We believe that our method *can* be extended to recover Sağlam and Tardos's [130] tradeoff (in the constant error probability regime), but with a more "direct" proof that avoids some technical difficulties arising from their round-elimination technique. It is still open whether EqualityTesting can be solved in r rounds with precisely $O(Ek^{1/r})$ communication and error probability $2^{-E} < 2^{-k}$. Our algorithms match this lower bound only when $r = O(1)$ or $r = \Omega(\log k)$, or for

[9] We sketch how the encoding of h works, for completeness. First, pick a function $h' : U \mapsto [O(k^2)]$ that is collision-free on A. Fredman et al. [66] proved that a function of the form $h'(x) = (ax \mod p) \mod O(k^2)$ works with constant probability, where $p = \Omega(k^2 \log|U|)$ is prime and $a \in [0, p)$ is random. Pick another function $h_* : [O(k^2)] \mapsto [k]$ that has at most twice the expected number of collisions on A, namely $2 \cdot \binom{k}{2}/k < k$, and partition A into k buckets $A_j = A \cap h_*^{-1}(j)$. The sizes $|A_0|, |A_1|, \ldots, |A_{k-1}|$ can be encoded with $2k$ bits. We now pick $O(\log k)$ pairwise independent hash functions $h_1, h_2, \ldots, h_{O(\log k)} : [O(k^2)] \mapsto [O(k^2)]$. For each bucket A_j, we define $h_{(j)}$ to be the function with the minimum i for which $h_{(j)}(x) = h_i(x) \mod |A_j|^2$ is injective on A_j. In order to encode which function $h_{(j)}$ is (given that $h_1, \ldots, h_{O(\log k)}$ are fixed and that $|A_j|$ is known), we simply need to write i in unary, i.e., using the bit-string $0^{i-1}1$. This takes less than 2 bits per j in expectation since each h_i is collision-free on A_j with probability at least 1/2. Combining $h', h_*, |A_0|, \ldots, |A_{k-1}|$ and $h_{(0)}, \ldots, h_{(k-1)}$ into a single injective function from $U \mapsto [O(k)]$ is straightforward, and done exactly as in [66]. By marking which elements in this range are actually used ($O(k)$ more bits), we can generate the perfect $h : U \mapsto [k]$ whose range has size precisely k. Encoding h' takes $O(\log k + \log\log|U|)$ bits and encoding h_* takes $O(\log k)$ bits. The distribution $|A_0|, \ldots, |A_{k-1}|$ can be encoded with $2k$ bits. The functions $h_1, \ldots, h_{O(\log k)}$ can be encoded in $O(\log^2 k)$ bits, and the functions $h_{(0)}, \ldots, h_{(k-1)}$ with less than $2k$ bits in expectation.

any r when solving the easier ExistsEqual problem.

We developed some CONGEST algorithms for triangle enumeration that employ two-party SetIntersection protocols. It is known that this strategy is suboptimal when $\Delta \gg n^{1/3}$ [27, 26]. However, for the *local* triangle enumeration problem[10], our $O(\Delta/\log n + \log\log\Delta)$ algorithm is optimal [89] for every $\Delta = \Omega(\log n \log\log\log n)$. Whether there are faster algorithms for triangle *detection*[11] is an intriguing open problem. It is known that 1-round LOCAL algorithms must send messages of $\Omega(\Delta \log n)$ bits deterministically [4] or $\Omega(\Delta)$ bits randomized [64]. Even for 2-round triangle detection algorithms, there are no nontrivial communication lower bounds known.

[10] Every triangle must be reported by one of its three constituent vertices.

[11] At least one vertex must announce there is a triangle; there is no obligation to list them all.

CHAPTER 3

Generalized Matching

3.1 Introduction

Many combinatorial optimization problems are known to be reducible to computing optimal matchings in non-bipartite graphs [59, 60]. These problems include computing b-matchings, f-factors, f-edge covers, T-joins, undirected shortest paths (with no negative cycles), and bidirected flows, see [105, 72, 132, 61]. These problems have been investigated heavily since Tutte's work in the 1950s [135, 123]. However, the existing reductions to graph matching are often inadequate: they blow up the size of the input [105], use auxiliary space [69], or piggyback on specific matching algorithms [69] like the Micali-Vazirani algorithm [110, 137, 138]. Moreover, most existing reductions destroy the dual structure of optimal solutions and are therefore not *approximation preserving*.

In this paper we design algorithms for computing f-matchings and f-edge covers (both defined below) in a direct fashion, or through efficient, approximation-preserving reductions. Because our algorithms are based on the LP formulations of these problems (in contrast to approaches using *shortest* augmenting walks [69, 110, 137, 138]), they easily adapt to *weighted* and *approximate* variants of the problems. Let us define these problems formally. Let $G = (V, E)$ be a graph that possibly contains parallel edges and self loops. For any subset F of edges, we use $\deg_F(v)$ to indicate the degree of vertex v in the subgraph induced by F. Notice that each self-loop on v that is in F contributes 2 to this degree. We use n to denote the number of vertices and m to denote the number of edges, counting multiplicities. We define f-matching and f-edge covers as follows.

f-matching An f-matching is a subset $F \subseteq E$ such that $\deg_F(v) \leq f(v)$. F is *perfect* if the degree constraints hold with equality. In this case it is also called an *f-factor*.

f-edge cover An f-edge cover is a subset $F \subseteq E$ such that $\deg_F(v) \geq f(v)$. It is *perfect* if all degree constraints hold with equality.

The *maximum weight f-matching problem* asks that, given a graph $G = (V, E)$ and a weight function w on E, to find an f-matching F that maximize $\sum_{e \in F} w(e)$. Similarly, the *minimum weight f-edge cover problem* and the *minimum weight f-factor problem* ask for an f-edge cover and an f-factor that minimize their respective weight.

For these three problems, we can assume, without lost of generality, that all the weights are nonnegative for different reasons. For maximum weight f-matching, it is safe to ignore any negative weight edges as discarding negative weight edges from F can only improve the solution. For minimum weight f-edge cover, any optimum solution must include the set of all negative weight edges. Hence we can include them into the solution and update the degree constraint accordingly. For minimum weight f-factor, since all f-factors are of the same size, we can translate all the weights by $-\min_{e \in E} w(e)$ without changing the optimal solution so that the resulting graph has only nonnegative weights.

Classic Reductions. The classical reduction from f-matching to standard graph matching uses the *b-matching* problem as a stepping stone. A b-matching is a function $x : E \to \mathbb{Z}_{\geq 0}$ (where $x(e)$ indicates *how many* copies of e are in the matching) such that $\sum_{e \in \delta(v)} x(e) \leq b(v)$, i.e., the number of matched edges incident to v, counting multiplicity, is at most $b(v)$. The maximum weight f-matching problem on $G = (V, E, w)$ can be reduced to b-matching by subdividing each edge $e = (u, v) \in E$ into a path (u, u_e, v_e, v). Here u_e, v_e are new vertices. We set the weight of the new edges to be $w(u, u_e) = w(v_e, v) = w(u, v) + W$ and $w(u_e, v_e) = 2W$, where W is the maximum weight in the original graph. The capacity function b is given by $b(u_e) = b(v_e) = 1$ for the new vertices and $b(u) = f(u)$ for the original vertices.

To see this reduction correctly reduces the maximum weight f-matching problem to the maximum weight b-matching problem, we first notice that if the original graph has an f-matching M of weight W^*, then the new graph must contain a b-matching of weight at least $2W^* + 2Wm$, where m is the number of edges in the original graph: for every unmatched edge (u, v), we take the edge (u_e, v_e) into the b-matching and leave the two edges (u, u_e) and (v, v_e) unmatched. Otherwise we take only the edges (u, u_e) and (v, v_e). For the other direction, we notice that if M is a maximum weight b-matching in the new graph, for each subdivision of original edge (u, v), we can assume without lost of generality that M' must either take only the middle edge

(u_e, v_e), or the two side edges (u, u_e) and (v, v_e): This is because by degree constraint on u_e and v_e, taking the middle edge will prevent the b-matching from taking any of the side edges, and vice versa. Moreover, we cannot take only one of the side edges since in that case we can swap it for the middle edge without decreasing the total weight. Therefore, a b-matching of weight $2W^* + 2mW$ must also correspond to an f-matching of weight W^* of the original graph.

This reduction blows up the number of vertices to $O(m)$ and is not *approximation preserving*. The b-matching problem is easily reduced to standard matching by replicating each vertex u $b(u)$ times, and replacing each edge (u, v) with a bipartite $b(u)\times b(v)$ clique on its endpoints' replicas. This step of the reduction is approximation preserving, but blows up the number of vertices and edges. Both reductions together reduce f-matching to a graph matching problem on $O(m)$ vertices and $O(f_{\max}m)$ edges. Gabow [69] gave a method for solving f-matching in $O(m\sqrt{f(V)})$ time using black-box calls to single iterations of the Micali-Vazirani [110, 137, 138] algorithm.

Observe that f-matching and f-edge cover are *complementary* problems: if C is an f_C-edge cover, the complementary edge set $F = E\backslash C$ is necessarily an f_F-matching, where $f_F(v) = \deg(v) - f_C(v)$. Complementarity implies that any polynomial-time algorithm for one problem solves the other in polynomial time, but it says nothing about the precise complexity of solving them exactly or approximately. Indeed, this phenomenon is very well known in the realm of NP-complete problems. For example, Maximum Independent Set and Minimum Vertex Cover are complementary problems, but have completely different approximation profiles: Minimum Vertex Cover has a well-known polynomial time 2-approximation algorithm, while it is NP-hard to approximate Maximum Independent Set within $n^{1-\epsilon}$ for any $\epsilon > 0$ [84, 145]. Gabow's $O(m\sqrt{f_F(V)})$ cardinality f_F-matching algorithm [69] implies that f_C-edge cover is computed in $O(m\sqrt{2m - f_C(V)}) = O(m^{3/2})$ time, and says nothing about the approximability of f_C-edge cover. As far as we are aware, the fastest approximation algorithms for f_C-edge cover (see [97]) treat it as a general weighted Set Cover problem on 2-element sets. Chvátal's analysis [39] shows the greedy algorithm is an $H(2)$-approximation, where $H(2) = 3/2$ is the 2nd harmonic number.

Our interest in the *approximate* f-edge cover problem is inspired by a new application to anonymizing data in environments where users have different privacy demands; see [97, 38, 98]. Here the data records correspond to edges and the privacy demand of v is measured by $f(v)$; the goal is to anonymize as few records to satisfy everyone's privacy demands.

New Results. We give new algorithms for computing f-matchings and f-edge covers approximately and exactly.

- We give an $O_\epsilon(m)$-time $(1-\epsilon)$-approximation algorithm for maximum weight f-matching problem. The algorithm generalizes the $(1-\epsilon)$-approximate maximum weight matching algorithm by Duan and Pettie [55] and improves on the $O(f(V)(m + n \log n))$ running time of Gabow [71]. The main technical contribution is the application of *relaxed complementary slackness* [74, 75, 55] on f-matchings, and a new version of DFS-based search procedure algorithm for looking for a *maximal* set of *edge-disjoint* augmenting paths in linear time.

- We show that a folklore reduction from minimum weight 1-edge cover to maximum weight 1-matching (matching) is approximation-preserving, in the sense that any $(1-\epsilon)$-approximation for matching gives a $(1+\epsilon)$-approximation for edge cover. This implies that 1-edge cover can be $(1+\epsilon)$-approximated in $O_\epsilon(m)$ time [55], and that one can apply any number of simple and practical algorithms [53, 120, 55] to approximate 1-edge cover. This simple reduction does *not* extend to f-matchings/f-edge covers when f is arbitrary.

- We give an $O_\epsilon(m)$-time $(1+\epsilon)$-approximation algorithm for weighted f_C-edge cover, for any f_C. Our algorithm follows from two results, both of which are somewhat surprising. First, any approximate weighted f_F-matching algorithm *that reports a* $(1\pm\epsilon)$*-optimal dual solution* can be transformed into a $(1+O(\epsilon))$-approximate weighted f_C-edge cover algorithm. Second, such an f_F-matching algorithm exists, and its running time is $O_\epsilon(m)$. The first claim is clearly false if we drop the approximate dual solution requirement (for the same reason that an $O(1)$-approximate vertex cover does not translate into an $O(1)$-approximate maximum independent set), and the second is surprising because the running time is independent of the demand function f_F and the magnitude of the edge weights.

- As corollaries of these reductions, we obtain a new exact algorithm for minimum cardinality f_C-edge cover running in $O(m\sqrt{f_C(V)})$ time, rather than $O(m^{3/2})$ time ([69]), and a direct algorithm for cardinality f_F-matching that runs in $O(m\sqrt{f_F(V)})$ time, without reduction [69] to the Micali-Vazirani algorithm [110, 137, 138].

The blossom structure and LP characterization of b-matching is considerably simpler than the corresponding blossoms/LPs for f-matching and f-edge cover. In

the interest of simplicity, one might want efficient code that solves (approximate) b-matching directly, without viewing it as a special case of the f-matching problem.[1] We do not know of such a direct algorithm. Indeed, the structure of b-matching blossoms seems to rely on strict complementary slackness, and is *incompatible* with our main technical tool, relaxed complementary slackness.[2] Thus, for somewhat technical reasons, we are forced to solve approximation b-matching using more sophisticated f-matching tools.

Comparison to Previous Results. Our linear time $(1-\epsilon)$-approximation algorithm for maximum weight f-matching can be seen as a direct generalization of the Duan-Pettie algorithm for approximate maximum weight matching (1-matching) [55]. The key technical ingredient is the generalization of *relaxed complementary slackness*, see [74, 75, 55], to f-matching, and a corresponding implementation of Edmonds' Search with relaxed complementary slackness. The former relies heavily on the ideas (blossoms, augmenting walks) defined in [71]. Our implementation of Edmonds' search involves finding augmenting walks in batches. The procedure of [75, §8] for matching finds a maximal set of vertex-disjoint augmenting paths. We develop a corresponding procedure that finds a maximal set of edge-disjoint augmenting walks *and* cycles. Including alternating cycles in the output allows us to conduct the search in linear time, and keep the search more organized and tree-structured.[3]

Organization. In Section 3.2 we give an introduction to the LP-formulation of generalized matching problems and Gabow's formulation [71] for their blossoms and augmenting walks. In Section 3.3.1 we show that a folklore reduction from 1-edge cover to 1-matching is approximation-preserving and in Section 3.3.2 we reduce approximate f-edge cover to approximate f-matching. In Section 3.4 we give an $O(Wm\epsilon^{-1})$-time algorithm for $(1-\epsilon)$-approximate f-matching in graphs with weights in $[0, W]$ and then speed it up to $O(m\epsilon^{-1}\log\epsilon^{-1})$, independent of the weight function. Section 3.5 gives a linear time algorithm to compute a maximal set of augmenting walks and alternating cycles; cf. [75, §8].

[1] The b-matching problem can be regarded as an f-matching problem on a multigraph in which there is implicitly an infinite supply of each edge.

[2] Using relaxed complementary slackness, matched and unmatched edges have *different* eligibility criteria (to be included in augmenting paths and blossoms) whereas b-matching blossoms require that all copies of an edge—matched and unmatched alike—are all eligible or all ineligible.

[3] These issues only arise when finding augmenting paths in batches, not one-at-a-time [71], and when the problem is f-matching, not matching.

3.2 Basis of f-Matching and f-Edge Cover

This section reviews basic algorithmic concepts from matching theory and their generalizations to the f-matching and f-edge cover problems, e.g., LPs, blossoms, and augmenting walks. These ideas lay the foundation for generalizing the Duan-Pettie algorithm [55] for Approximate Maximum Weight Matching to Approximate Maximum Weight f-Matching and Approximate Minimum Weight f-Edge Cover.

Notation. The input is a multigraph $G = (V, E)$ with a *nonnegative* weight function $w : E \mapsto \mathbb{R}_{\geq 0}$. For any vertex v, define $\delta(v)$ and $\delta_0(v)$ be the set of non-loop edges and self-loops, respectively, incident on v. For $S \subseteq V$, let $\delta(S)$ and $\gamma(S)$ be the sets of edges with exactly one endpoint and both endpoints in S, respectively, so $\delta_0(v) \subseteq \gamma(S)$ if $v \in S$. For $T \subseteq E$, $\delta_T(S)$ denotes the intersection of $\delta(S)$ and T. By definition, $\deg_T(S) = |\delta_T(S)|$.

3.2.1 LP Formulation

The maximum weight f-matching problem can be expressed as maximizing $\sum_{e \in E} w(e)x(e)$, subject to the following constraints:

$$\begin{aligned}
&\sum_{e \in \delta(v)} x(e) + \sum_{e \in \delta_0(v)} 2x(e) \leq f(v), \text{ for all } v \in V, \\
&\sum_{e \in \gamma(B) \cup I} x(e) \leq \left\lfloor \frac{f(B) + |I|}{2} \right\rfloor, \text{ for all } B \subseteq V, I \subseteq \delta(B), \\
&0 \leq x(e) \leq 1, \text{ for all } e \in E.
\end{aligned} \tag{3.1}$$

Here, the blossom constraint $\sum_{e \in \gamma(B) \cup I} x(e) \leq \left\lfloor \frac{f(B)+|I|}{2} \right\rfloor$ is a generalization of blossom constraint $\sum_{e \in \gamma(B)} x(e) \leq \left\lfloor \frac{|B|}{2} \right\rfloor$ in ordinary matching. The reason that we have a subset I of incident edges in the sum is that the subset allows us to distinguish between matched edges that have both endpoints inside B with those with exactly one endpoint. Any basic feasible solution x of this LP is integral [132, §33], and can therefore be interpreted as a membership vector of an f-matching F. To certify (approximate) optimality of a solution, the algorithm works with the dual LP, which is:

$$\begin{aligned}
\text{minimize} \quad & \sum_{v \in V} f(v) y(v) + \sum_{B \subseteq V, I \subseteq \delta(B)} \left\lfloor \frac{f(B) + |I|}{2} \right\rfloor z(B, I) + \sum_{e} u(e), \\
\text{subject to} \quad & y z_F(e) + u(e) \geq w(e), \text{ for all } e \in E, \\
& y(v) \geq 0, z(B, I) \geq 0, u(e) \geq 0.
\end{aligned} \tag{3.2}$$

Here the aggregated dual $yz_F : E \mapsto \mathbb{R}_{\geq 0}$ is defined as:

$$yz_F(u, v) = y(u) + y(v) + \sum_{\substack{B, I : (u,v) \in \gamma(B) \cup I, \\ I \subseteq \delta(B))}} z(B, I).$$

Notice that here u can be equal to v when the edge is a self-loop. Unlike matching, each z-value here is associated with the combination of a vertex set B and a subset I of its incident edges.

The minimum weight f-edge cover problem can be expressed as minimizing $\sum_{e \in E} w(e)x(e)$, subject to:

$$\begin{aligned}
& \sum_{e \in \delta(v)} x(e) + \sum_{e \in \delta_0(v)} 2x(e) \geq f(v), \text{ for all } v \in V, \\
& \sum_{e \in \gamma(B) \cup (\delta(B) \setminus I)} x(e) \geq \left\lceil \frac{f(B) - |I|}{2} \right\rceil, \text{ for all } B \subseteq V \text{ and } I \subseteq \delta(B), \\
& 0 \leq x(e) \leq 1, \text{for all } e \in E.
\end{aligned} \tag{3.3}$$

With the dual program being:

$$\begin{aligned}
\text{maximize} \quad & \sum_{v \in V} f(v) y(v) + \sum_{B \subseteq V, I \subseteq \delta(B)} \left\lceil \frac{f(B) - |I|}{2} \right\rceil z(B, I) - \sum_{e \in E} u(e), \\
\text{subject to} \quad & y z_C(e) - u(e) \leq w(e), \text{ for all } e \in E, \\
& y(v) \geq 0, z(B, I) \geq 0, u(e) \geq 0,
\end{aligned} \tag{3.4}$$

where

$$yz_C(u, v) = y(u) + y(v) + \sum_{\substack{B, I : (u,v) \in \gamma(B) \cup (\delta(B) \setminus I) \\ I \subseteq \delta(B)}} z(B, I).^4$$

[4]We use yz_F and yz_C to denote the aggregated dual yz for f-matching and f-edge cover respectively. We will omit the subscript if it is clear from the context.

Both of our f-matching and f-edge cover algorithms maintain a dynamic *feasible* solution $F \subseteq E$ that satisfies the primal constraints following Gabow[71]. We call edges in F *matched* and all other edges *unmatched*, which is referred to as the *type* of an edge. A vertex v is *saturated* if $\deg_F(v) = f(v)$. It is *unsaturated/oversaturated* if $\deg_F(v)$ is smaller/greater than $f(v)$. Given an f-matching F, the *deficiency* $\text{def}(v)$ of a vertex v is defined as $\text{def}(v) = f(v) - \deg_F(v)$. Similarly, for an f-edge cover C, the *surplus* of a vertex is defined as $\text{surp}(v) = \deg_C(v) - f(v)$.

3.2.2 Blossoms

We follow Gabow's [71] definitions and terminology for f-matching blossoms, augmenting walks, etc. A *blossom* is a tuple $(B, E_B, \beta(B), \eta(B))$ where B is the vertex set, E_B is the edge set, $\beta(B) \in B$ is the *base vertex*, and $\eta(B) \subset \delta(\beta(B)) \cap \delta(B)$, $|\eta(B)| \leq 1$, is the *base edge set*, which may be empty. We often refer to the blossom by referring to its vertex set B. Blossoms can be defined inductively as follows.

Definition 3.1. *[71, Definition 4.2] A single vertex v forms a* trivial blossom*, or a* singleton*. Here $B = \{v\}$, $E_B = \emptyset$, $\beta(B) = v$, and $\eta(B) = \emptyset$.*

Inductively, let $B_0, B_1, \ldots, B_{l-1}$ be a sequence of disjoint singletons or nontrivial blossoms. Suppose there exists a closed walk $C_B = \{e_0, e_1, \ldots, e_{l-1}\} \subseteq E$ starting and ending with B_0 such that $e_i \in B_i \times B_{i+1 \pmod l}$. The vertex set $B = \bigcup_{i=0}^{l-1} B_i$ is identified with a blossom if the following are satisfied:

1. *Base Requirement: If B_0 is a singleton, the two edges incident to B_0 on C_B, i.e., e_0 and e_{l-1}, must both be matched or both be unmatched.*
2. *Alternation Requirement: Fix a $B_i, i \neq 0$. If B_i is a singleton, exactly one of e_{i-1} and e_i is matched. If B_i is a nontrivial blossom, then $\eta(B_i) \neq \emptyset$ and must be either $\{e_{i-1}\}$ or $\{e_i\}$.*

The edge set of the blossom B is $E_B = C_B \cup (\bigcup_{i=0}^{l-1} E_{B_i})$ and its base is $\beta(B) = \beta(B_0)$. If B_0 is not a singleton, $\eta(B) = \eta(B_0)$. If B_0 is a singleton, $\eta(B)$ may either be empty or contain one edge, which is in $\delta(B) \cap \delta(B_0)$ that is the opposite type of e_0 and e_{l-1}.

Blossoms are classified as *light/heavy* [71, p. 32]. If B_0 is a singleton, B is light/heavy if e_0 and e_{l-1} are both unmatched/matched. Otherwise, B is light/heavy if B_0 is light/heavy. Note that blossoms in the ordinary matching problem (1-matching) are always light, since no vertex is adjacent to 2 matched edges.

One purpose of blossoms is to identify parts of graph that can be contracted and treated *similar* to individual vertices when searching for augmenting walks. This is formalized by Lemma 3.2, which can be seen as a restatement of Lemma 4.4 from [71] for f-matchings.

Lemma 3.2. *Let v be an arbitrary vertex in B. There exists an even length alternating walk $P_0(v)$ (whose length could be 0) and an odd length alternating walk $P_1(v)$ from $\beta(B)$ to v using edges in E_B. Moreover, the terminal edge incident to $\beta(B)$, if it exists, must have a different type than the edge in $\eta(B)$, if any. In other words, this edge must be matched if B is heavy and unmatched if B is light.*

Proof. We prove this by induction. The base case is a blossom B consisting of singletons $\langle v_0, v_1, \dots, v_{l-1}\rangle$, where $v = v_i$ for some $0 \leq i < l$. Then one of the two walks $\langle v_0, v_1, \dots, v_i\rangle$ and $\langle v_0, v_{l-1}, v_{l-2} \dots, v_i\rangle$ must be odd and the other must be even.

Now for the inductive step: Consider the cycle $C_B = \langle B_0, e_0, B_1, \dots, e_{l-2}, B_{l-1}e_{l-1}, B_0\rangle$ where B_i's are singletons or contracted blossoms. Suppose the claim holds inductively for all nontrivial blossoms in $B_0, B_1, \dots, B_{l-1}$. Let v be an arbitrary vertex in B. We use $P_{B_i,j}(u)$ $(0 \leq i < l, j \in \{0,1\}, u \in B_i)$ to denote the walk $P_0(u)$ and $P_1(u)$ guaranteed in blossom B_i. There are two cases:

Case *1:* When v is contained in a singleton B_k. We examine the two walks $\widehat{P} = \langle B_0, e_0, B_1, e_1, \dots, e_{k-1}, B_k\rangle$ and $\widehat{P}' = \langle B_0, e_{l-1}, B_{l-1}, e_{l-2}, \dots, e_k, B_k\rangle$. Notice that $\widehat{P}$ and $\widehat{P}'$ are walks in the graph obtained by contracting all subblossoms $B_0, B_1, \dots, B_{l-1}$ of B. By the inductive hypothesis, we can extend $\widehat{P}$ and $\widehat{P}'$ to P and P' in the original graph G by replacing each B_i with the walk in the original graph connecting the endpoints of e_{i-1} and e_i of the appropriate parity. In particular, if e_{i-1} and e_i are of different types, we replace B_i with the even length walk guaranteed by the induction hypothesis. Otherwise, we replace it with the odd length walk. Notice that by the alternation requirement, one of P and P' must be odd and the other must be even.

Case *2:* When v is contained in a non-trivial blossom B_k, $0 \leq k < l$. Without loss of generality, $e_{k-1} = \eta(B_k)$. Consider the contracted walk $\widehat{P} = \langle e_0, e_1, \dots, e_{k-1}\rangle$. We extend $\widehat{P}$ to an alternating walk P in E_B terminating at e_{k-1} similar to Case 1. Then $P_0(v)$ and $P_1(v)$ are obtained by concatenating P with the alternating walk $P_{B_k,0}(v)$ or $P_{B_k,1}(v)$, whichever has the right parity.

Notice that in both cases, the *base requirement* in Definition 3.1 guarantees the starting edge of both alternating walks $P_1(v)$ and $P_0(v)$ alternates with the base edge $\eta(B)$. □

The main difference between blossoms in generalized matching problems and blossoms in ordinary matching is that $P_0(v)$ and $P_1(v)$ are *both* meaningful for finding augmenting walks or blossoms. In ordinary matching, since each vertex has at most 1 matched edge incident to it, an alternating walk enters the blossom at the base vertex via a matched edge and must leave with an unmatched edge. As a result the subwalk inside the blossom is always even. In generalized matching problems, this subwalk can be either even or odd, and may contain a cycle. In general, an alternating walk enters the blossom at the base edge and can leave the blossom at *any* nonbase edge.

Similar to ordinary matching algorithms, we *contract* blossoms in order to find augmenting structures to improve our f-matching. Contracting a blossom B means replacing B with a single vertex v with an f-value $f(v) = \sum_{v' \in B} f(v') - 2|M \cap E[B]|$. Here $E[B]$ is the set of edges induced by the vertex set B.

Next we extend to notion of *maturity* from [71, p. 43] to f-matching and f-edge cover. Let us focus on f-matching first. Due to complementary slackness, we can only assign a positive z-value for the pair (B, I) if it satisfies the constraint $|F \cap (\gamma(B) \cup I)| \leq \lfloor (f(B) + |I|)/2 \rfloor$ with equality. For ordinary matching, this requirement is implied by the combinatorial definition of blossoms. However, this is not the case for generalized matching, so we need a blossom to be *mature* to fulfill the complementary slackness property.

Definition 3.3 (Mature Blossom)**.** *A blossom is* mature *w.r.t an f-matching F if it satisfies the following:*

1. *Every vertex $v \in B \setminus \{\beta(B)\}$ is saturated.*
2. $\operatorname{def}(\beta(B)) = 0$ *or* 1*. If* $\operatorname{def}(\beta(B)) = 1$*, B must be a light blossom and $\eta(B) = \emptyset$; If* $\operatorname{def}(\beta(B)) = 0$*, $\eta(B) \neq \emptyset$.*

The algorithm only contracts and manipulates mature blossoms. The definition for maturity is motivated by the requirement that a blossom processed by the algorithm must satisfy the following two properties:

- Complementary slackness: A dual variable can be positive only if its primal constraint is satisfied with equality. In our algorithm, a blossom can have a positive z-value only if $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$, for a particular $I(B) \subseteq \delta(B)$ that we are going to define momentarily.
- Topology of augmenting walks: An augmenting walk in G can only start with an unmatched edge. As a result, an augmenting walk in the contracted graph must start with a singleton or an unsaturated light blossom. If a blossom is

unsaturated, it must be eligible to start an augmenting walk, and thus must be light.

According to Definition 3.3, a mature blossom cannot be both heavy and unsaturated. Now we show that a mature blossom satisfies its corresponding primal constraint with equality. To show this fact, we first define the I-set of a blossom B [71, p.44], which is the set $I(B)$ associated with blossom B for which we will assign a positive z-value, given by:

$$I(B) = \delta_F(B) \oplus \eta(B),$$

where $\oplus$ is the symmetric difference operator (XOR). All other subsets I of $\delta(B)$ will have $z(B, I) = 0$. If B is a mature blossom, then we have $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$

Lemma 3.4. *If an f-matching blossom B is mature, we have $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$.*

Proof. We first sketch the idea of the proof. Assume for simplicity that the deficiency is 0 for every vertex $v \in B$, i.e., there are exactly $f(v)$ matched edges incident to v, and every edge in $I(B)$ is matched. Then every matched edge $e \in F \cap \gamma(B)$ contributes 2 to $f(B)$, one for each endpoint, and every edge $e \in I(B) \cap F$ contributes 1 to $f(B)$ and 1 to $|I(B)|$. Thus we have:

$$2|F \cap (\gamma(B) \cup I(B))| = f(B) + |I(B)|.$$

Now we eliminate the assumption by a case analysis for the deficiency of $\beta(B)$. If $\text{def}(\beta(B)) = 0$, the assumption on deficiency holds, while all but possibly 1 edge in $I(B)$, namely the base edge, are matched. This makes $f(B) + |I(B)|$ at least $2|F \cap (\gamma(B) \cup I(B))|$ and at most $2|F \cap (\gamma(B) \cup I(B))| + 1$, and the equality $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$ follows.

When $\text{def}(\beta(B)) = 1$, since $\eta(B) = \emptyset$, the assumption that $I(B)$ only contains matched edges holds. Since exactly 1 of B's vertices has deficiency 1, we have:

$$2|F \cap (\gamma(B) \cup I(B))| + 1 = f(B) + |I(B)|$$

And the equality $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$ follows. □

We complete the discussion by giving the definition for maturity and the corresponding properties for mature blossoms in f-edge cover. The details are similar to

f-matching.

Definition 3.5 (Mature Blossom for f-edge cover)**.** *A blossom is* mature *w.r.t an f-edge cover F if it satisfies the following:*

1. *Every vertex $v \in B \setminus \{\beta(B)\}$ is saturated:* $\deg_F(v) = f(v)$.
2. $\mathrm{surp}(\beta(B)) = 0$ *or* 1. *If* $\mathrm{surp}(\beta(B)) = 1$, *$B$ must be a heavy blossom and $\eta(B) = \emptyset$; If* $\mathrm{surp}(\beta(B)) = 0$, *$\eta(B) \neq \emptyset$.*

Lemma 3.6. *If an f-edge-cover blosom B is mature, we have* $|F \cap (\gamma(B) \cup (\delta(B) \setminus I(B)))| = \left\lceil \frac{f(B)-|I(B)|}{2} \right\rceil$.

3.2.3 Augmenting/Reducing Walks

Augmenting walks are analagous to augmenting paths from ordinary matching. Complications arise from the fact that an f-matching blossom cannot be treated identically to a single vertex after it is contracted. For example, in Figure 2, the two edges (v_0, v_1) and (v_4, v_6) incident to blossom $\{v_1, v_2, v_4, v_5, v_3\}$ are of the same type, both before and after augmenting along the walk $\langle v_0, v_1, v_3, v_5, v_4, v_6 \rangle$. This can never happen in ordinary matching! Moreover, augmenting walks can begin and end at the same vertex and can visit the same vertex multiple times. Hence a naive contraction of a blossom into a single vertex loses key information about the internal structure of blossoms. Definition 3.7, taken from Gabow [71, p.28, p.44] characterizes when a walk in the contracted graph can be extended to an augmenting walk.

Definition 3.7. *Let $\widehat{G}$ be the graph obtained from G by contracting a laminar set Ω of blossoms. Let $\widehat{P} = \langle B_0, e_0, B_1, e_1, \ldots, B_{l-1}, e_{l-1}, B_l \rangle$ be a walk in $\widehat{G}$. Here $\{e_i\}$ are edges and $\{B_i\}$ are nontrivial blossoms or singletons, with $e_i \in B_i \times B_{i+1}$ for all $0 \leq i < l$. We say $\widehat{P}$ is an* augmenting walk *with respect to the f-matching F if the following requirements are satisfied:*

1. *Terminal Vertices Requirement: The terminals B_0 and B_l must be unsaturated singletons or unsaturated light nontrivial blossoms. If P is a closed walk ($B_0 = B_l$), B_0 must be a singleton and* $\mathrm{def}(\beta(B_0)) \geq 2$.
2. *Terminal Edges Requirement: If the terminal vertex B_0 (B_l) is a singleton, the incident terminal edge e_0 (e_{l-1}) must be unmatched. Otherwise it can be either matched or unmatched.*

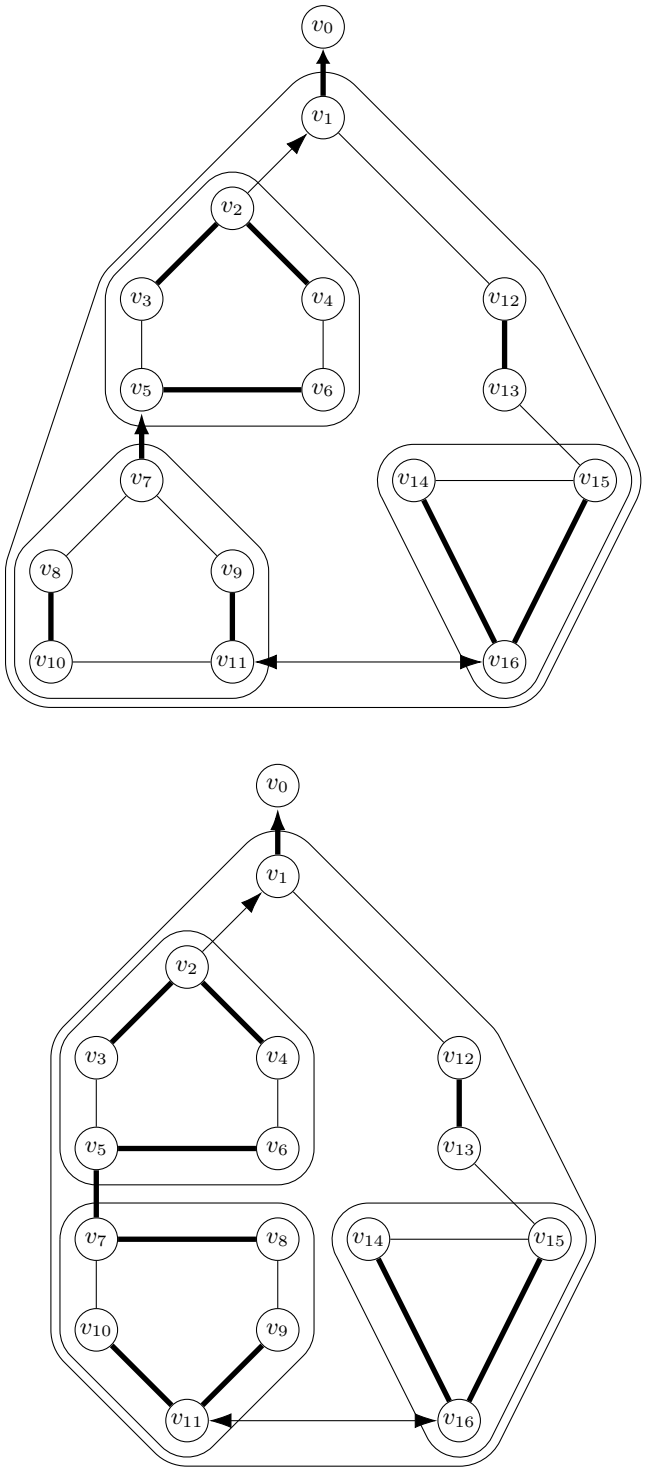


Figure 3.1: Two examples of contractible blossoms: Bold edges are matched and thin ones are unmatched. Blossoms are circled with a border. Base edges are represented with arrow pointing away from the blossom.

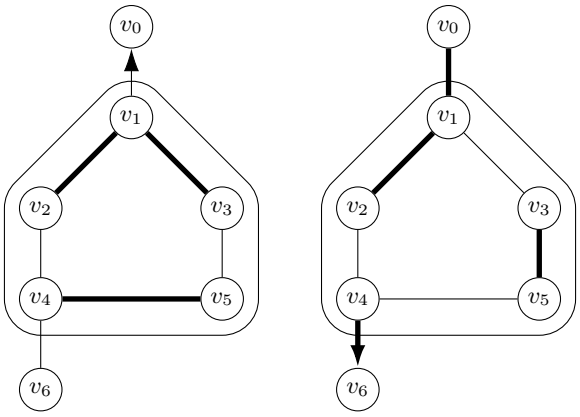


Figure 3.2: An example for how a blossom changes with an augmentation: here the augmenting walk is $\langle v_0, v_1, v_3, v_5, v_4, v_6\rangle$. Notice that after rematching, the base edge of the blossom changes from (v_0, v_1) to (v_4, u_6), and the blossom turns from a heavy blossom to a light one.

3. *Alternation Requirement: Let $B_i, 0 < i < l$, be an internal blossom. If B_i is a singleton, exactly one of e_{i-1} and e_i is matched. If B_i is a nontrivial blossom, $\eta(B_i) \neq \emptyset$ and must be one of $\{e_{i-1}\}$ or $\{e_i\}$.*

A natural consequence of the above definition is that an augmenting walk $\widehat{P}$ in $\widehat{G}$ can be extended to an augmenting walk P in G. This is proved exactly as in Lemma 3.2. We call P the *preimage* of $\widehat{P}$ in G and $\widehat{P}$ the image of P in $\widehat{G}$.

Definition 3.8. *Let $\widehat{P}$ be an augmenting walk in $\widehat{G}$. An* augmentation *along $\widehat{P}$ makes the following changes to F and Ω.*

1. *Let P be the preimage of $\widehat{P}$ in G. Update F to $F \oplus P$.*
2. *If $B \in \Omega$ is a blossom intersecting P, we set $\eta(B) \leftarrow (P \cap \delta(B)) \setminus \eta(B)$ and set $\beta(B)$ to the vertex in B that is incident to the edge in $\eta(B)$. Notice that $|P \cap \delta(B)| = 1$ or 2, and in the case when $|P \cap \delta(B)| = 1$, we must have $\eta(B) = \emptyset$.*

Some remarks can be made here regarding connection to augmenting walks and mature blossoms.

- A blossom that is not mature may contain an augmenting walk. Specifically, suppose B is light and unsaturated. If any nonbase vertex $v \neq \beta(B)$ in B is also unsaturated, the odd length alternating walk from $\beta(B)$ to v satisfies the definition of an augmenting walk. Alternatively, if $\beta(B)$ has deficiency of 2 or more, the odd length alternating walk from $\beta(B)$ to $\beta(B)$ is also augmenting.

For these reasons, the algorithm is designed such that immature blossoms are never contracted.

- Augmentation never destroys maturity. In particular, it never creates an unsaturated heavy blossom. As a result, all blossoms we maintain stay mature throughout the entirety of the algorithm.

In f-edge cover, the corresponding notion is called *reducing walk*. The definition of reducing walk can be naturally obtained from Definition 3.7 while replacing "unsaturated", "deficiency", and "light" with "oversaturated", "surplus", and "heavy", and exchanging "matched" and "unmatched". It is also worth pointing out that if an f-matching F and an f'-edge cover F' are *complement to* each other, i.e., $F' = E \setminus F$ and $f(v)+f'(v) = \deg(v)$, and they have the same blossom set Ω, then an augmenting walk $\widehat{P}$ for F is also a reducing walk for F'.

3.2.4 Complementary Slackness

To characterize an (approximately) optimal solution, we maintain dual functions: $y : V \mapsto \mathbb{R}_{\geq 0}$ and $z : 2^V \mapsto \mathbb{R}_{\geq 0}$. Here $z(B)$ is short for $z(B, I(B))$. We do not explicitly maintain the edge dual $u : E \mapsto \mathbb{R}_{\geq 0}$ since its minimizing value can be explicitly given by $u(e) = \max\{w(e) - yz(e), 0\}$. For f-matching F, the following property characterizes an approximate maximum weight f-matching:

Property 3.9 (Approximate Complementary Slackness for f-matching)**.** *Let $\delta_1, \delta_2 \geq 0$ be nonnegative parameters. We say an f-matching F, duals y, z, and the set of blossoms Ω satisfies (δ_1, δ_2)-approximate complementary slackness if the following hold:*

1. *Approximate Domination. For each unmatched edge $e \in E \setminus F$, $yz(e) \geq w(e) - \delta_1$.*
2. *Approximate Tightness. For each matched edge $e \in F$, $yz(e) \leq w(e) + \delta_2$.*
3. *Blossom Maturity. For each blossom $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$.*
4. *Unsaturated Vertices' Duals. For each unsaturated vertex v, $y(v) = 0$.*

Lemma 3.10. *Let F be an f-matching in G along with duals y, z and let F^* be the maximum weight f-matching. If F, Ω, y, z satisfy Property 3.9 with parameters δ_1 and δ_2, we have*

$$w(F) \geq w(F^*) - \delta_1|F^*| - \delta_2|F|.$$

Proof. We first define $u : E \mapsto \mathbb{R}$ as

$$u(e) = \begin{cases} w(e) - yz(e) + \delta_2, & \text{if } e \in F. \\ 0, & \text{otherwise.} \end{cases}$$

From approximate tightness, we have $u(e) \geq 0$ for all $e \in E$. Therefore, $yz(e)+u(e) \geq w(e) - \delta_1$ for all $e \in E$ and $yz(e) + u(e) = w(e) + \delta_2$ for all $e \in F$. This gives the following:

$$\begin{aligned} w(F) &= \sum_{e \in F} w(e) = \sum_{e \in F} (yz(e) + u(e) - \delta_2) \\ &= \sum_{v \in V} \deg_F(v) y(v) + \sum_{B \in \Omega} |F \cap (\gamma(B) \cup I(B))| z(B) + \sum_{e \in F} u(e) - |F|\delta_2 \end{aligned}$$

By Property 3.9 (Unsaturated Vertices' Duals, Blossom Maturity, and the definition of u), this is equal to

$$\begin{aligned} &= \sum_{v \in V} f(v) y(v) + \sum_{B \in \Omega} \left\lfloor \frac{f(B) + |I(B)|}{2} \right\rfloor z(B) + \sum_{e \in E} u(e) - |F|\delta_2 \\ &\geq \sum_{v \in V} \deg_{F^*}(v) y(v) + \sum_{B \in \Omega} |F^* \cap (\gamma(B) \cup I(B))| z(B) + \sum_{e \in F^*} u(e) - |F|\delta_2 \\ &= \sum_{e \in F^*} (yz(e) + u(e)) - |F|\delta_2 \\ &\geq \sum_{e \in F^*} (w(e) - \delta_1) - |F|\delta_2 = w(F^*) - |F^*|\delta_1 - |F|\delta_2. \end{aligned}$$

□

We can easily extend the proof of Lemma 3.10 to show that if we have multiplicative errors for approximate domination/tightness, F is an approximately optimal solution. Formally, if we have the following multiplicative version of Property 3.9:

Property 3.11 (Approximate Complementary Slackness for f-matching with Multiplicative Error)**.** *Let $0 \leq \epsilon_1, \epsilon_2 < 1$ be nonnegative parameters. We say an f-matching F, duals y, z, and the set of blossoms Ω satisfies (ϵ_1, ϵ_2)-multiplicative approximate complementary slackness if it satisfies Property 3.9(3,4), with Property 3.9(1,2) being replaced with:*

1. *Approximate Domination. For each unmatched edge $e \in E \setminus F$, $yz(e) \geq (1 - \epsilon_1)w(e)$.*

2. *Approximate Tightness. For each matched edge* $e \in F$, $yz(e) \leq (1+\epsilon_2)w(e)$.

We can show the following:

Lemma 3.12. *Let* F *be an* f*-matching in* G *along with duals* y, z *and let* F^* *be the maximum weight* f*-matching. If* F, Ω, y, z *satisfy Property 3.11 with parameters* ϵ_1 *and* ϵ_2*, we have*

$$w(F) \geq (1-\epsilon_1)(1+\epsilon_2)^{-1}w(F^*)$$

We also give the corresponding theorems for f-edge covers:

Property 3.13 (Approximate Complementary Slackness for f-edge cover)**.** *Let* $\delta_1, \delta_2 \geq 0$ *be positive parameters. We say an* f*-edge cover* C*, with duals* y*,* z *and blossom family* Ω *satisfies the* (δ_1, δ_2)*-approximate complementary slackness if the following requirements holds:*

1. *Approximate Domination. For each unmatched edge* $e \in E \setminus C$, $yz_C(e) \leq w(e) + \delta_1$.
2. *Approximate Tightness. For each matched edge* $e \in C$, $yz_C(e) \geq w(e) - \delta_2$.
3. *Blossom Maturity. For each blossom* $B \in \Omega$, $|C \cap (\gamma(B) \cup (\delta(B) \setminus I_C(B)))| = \left\lceil \frac{f(B)-|I_C(B)|}{2} \right\rceil$.
4. *Oversaturated Vertices' Duals. For each oversaturated vertex* v, $y(v) = 0$.

Property 3.14 (Approximate Complementary Slackness for f-edge cover with Multiplicative Error)**.** *Let* $0 \leq \epsilon_1, \epsilon_2 < 1$ *be positive parameters. We say an* f*-edge cover* C*, with duals* y*,* z *and blossom family* Ω *satisfies the* (ϵ_1, ϵ_2)*-approximate complementary slackness if it satisfies Property 3.13(3,4), with Property 3.13(1,2) being replaced with:*

1. *Approximate Domination. For each unmatched edge* $e \in E \setminus C$, $yz_C(e) \leq (1+\epsilon_1)w(e)$.
2. *Approximate Tightness. For each matched edge* $e \in C$, $yz_C(e) \geq (1-\epsilon_2)w(e)$.

Recall that we are using the aggregated duals yz_C for f-edge cover:

$$yz_C(u,v) = y(u) + y(v) + \sum_{B:(u,v)\in\gamma(B)\cup(\delta(B)\setminus I_C(B))} z(B)$$

Lemma 3.15. *Let* C *be an* f*-edge cover with duals* y, z, Ω *satisfying Property 3.13 with parameters* δ_1 *and* δ_2*, and let* C^* *be the minimum weight* f*-edge cover. We have* $w(C) \leq w(C^*) + \delta_1|C^*| + \delta_2|C|$.

Lemma 3.16. *Let C be an f-edge cover with duals y, z, Ω satisfying Property 3.14 with parameters ϵ_1 and ϵ_2, and let C^* be the minimum weight f-edge cover. We have $w(C) \leq (1+\epsilon_1)(1-\epsilon_2)^{-1}w(C^*)$.*

3.3 Connection Between f-Matchings and f-Edge Covers

The classical approach to solve the f-edge cover problem is to reduce it to f-matching. Specifically, looking for a minimum weight f_C-edge cover C for some function f_C can be seen as choosing edges that are *not* in C, which is a maximum weight f_F-matching where $f_F(u) = \deg(u) - f_C(u)$.

The main drawback of this reduction is that it yields inefficient algorithms. For example, Gabow's algorithms [71] for solving maximum weight f_F-matching scales linearly with $f_F(V)$, which makes it undesirable when f_C is small. Even when $f_C(V) = O(n)$, Gabow's algorithm runs in $O(m^2 + mn \log n)$ time. Moreover, this reduction is not approximation-preserving. In other words, the complement of an *arbitrary* $(1-\epsilon)$-approximate maximum weight f_F-matching is not guaranteed to be a $(1+\epsilon)$-approximate f_C-edge cover.

In this section we establish two results: First we prove that a folklore reduction from 1-edge cover to matching in nonnegative weight graphs is approximation preserving. This allows us to use an efficient approximate matching algorithm for ordinary matching, such as [55], to solve the weighted 1-edge cover problem. Then we establish the connection between approximate f_F-matching and approximate f_C-edge cover using approximate complementary slackness from the previous section. This will give a $(1+\epsilon)$-approximate minimum weight f-edge cover algorithm from our $(1-\epsilon)$ approximate maximum weight f-matching algorithm.

3.3.1 Approximate Preserving Reduction from 1-Edge Cover to 1-Matchings

The edge cover problem is a special case of f-edge cover where f is 1 everywhere. The minimum weight edge cover problem is reducible to maximum weight matching, simply by reweighting edges [132]. The reduction is as follows: Let $e(v)$ be any edge with minimum weight incident to v and let $\mu(v) = w(e(v))$. Define a new weight

function w' as follows

$$w'(u,v) = \mu(u) + \mu(v) - w(u,v).$$

Schrijver [132, §27] showed the following theorem:

Theorem 3.17. *Let M^* be a maximum weight matching with respect to a nonnegative weight function w', and $C = M^* \cup \{e(v) : v \in V \setminus V(M)\}$. Then C is a minimum weight edge cover with respect to weight function w.*

We show this reduction is also approximation preserving. Recall that the generally weighted versions of these problems are reducible to the *non-negatively* weighted versions in linear time.

Theorem 3.18. *Let M' be a $(1-\epsilon)$-maximum weight matching with respect to non-negative weight function w', and $C' = M' \cup \{e(v) : v \in V \setminus V(M')\}$. Then C' is a $(1+\epsilon)$-minimum weight edge cover with respect to weight function w.*

Proof. Let C^* and M^* be the optimal edge cover and matching defined previously. By construction, we have

$$\begin{aligned} w(C') &= w(M') + \mu(V \setminus V(M')) \\ &= \mu(V(M')) - w'(M') + \mu(V \setminus V(M')) \\ &= \mu(V) - w'(M') \end{aligned}$$

Similarly, we have $w(C^*) = \mu(V) - w'(M^*)$. Then

$$\begin{aligned} w(C') =& \mu(V) - w'(M) \\ \leq& \mu(V) - (1-\epsilon)w'(M^*) \\ =& w(C^*) + \epsilon w'(M^*) \\ \leq& w(C^*) + \epsilon w(C^*) \\ =& (1+\epsilon)w(C^*). \end{aligned}$$

The second to last inequality holds because $M^* \subseteq C^*$ and, by definition, $w'(u,v) = \mu(u) + \mu(v) - w(u,v) \leq 2w(u,v) - w(u,v) = w(u,v)$. □

The reduction does not naturally extend to f-edge cover. In the next section we will show how to obtain a $(1+\epsilon)$-approximate f-edge cover algorithm from a $(1-\epsilon)$-approximate f-matching within the primal-dual framework.

3.3.2 From f-Edge Cover to f-Matching

We show that a primal-dual algorithm computing a $(1-\epsilon)$-approximate f-matching can be used to compute a $(1+\epsilon)$-approximate f-edge cover. In particular, we show that if we have an f'-matching F with blossoms Ω and duals y, z satisfying Property 3.9, and an f-edge cover C that is F's complement, then the same blossom set Ω and duals y, z can be also used to certify Property 3.13 for C with a same set of parameters. This is formally stated in the following lemma:

Lemma 3.19. *If the duals y, z, Ω and an f'-matching F satisfy Property 3.9 with parameters δ_1', δ_2', then the same duals y, z, Ω and the complementary f-edge cover $C = E \setminus F$ satisfies Property 3.13 with parameters $\delta_1 = \delta_2'$ and $\delta_2 = \delta_1'$.*

Proof. It is easy to see a vertex is oversatured in an f-edge cover if and only if it is unsaturated in its complementary f-matchings. Therefore, Property 3.13(4) (Oversaturated Vertices' Duals) and Property 3.9(4) (Unsaturated Vertices' Duals) are equivalent to each other.

To show Property 3.13(1) is equivalent to Property 3.9(2), and Property 3.13(2) is equivalent to Property 3.9(1), it suffices to show that the function yz_F for f'-matching F agrees with the function yz_C for its complementary f-edge cover C. Recall that

$$yz_C(u,v) = y(u) + y(v) + \sum_{B:(u,v)\in\gamma(B)\cup(\delta(B)\setminus I_C(B))} z(B),$$
$$yz_F(u,v) = y(u) + y(v) + \sum_{B:(u,v)\in\gamma(B)\cup I_F(B)} z(B).$$

Here I_C and I_F refer to the I-sets of a blossom with respect to the f-edge cover C and the f'-matching F. This reduces to showing that $I_F(B) = \delta(B) \setminus I_C(B)$:

$$\begin{aligned} I_F(B) &= \eta(B) \oplus \delta_F(B) = \eta(B) \oplus (\delta(B) \oplus \delta_C(B)) \\ &= \delta(B) \oplus (\eta(B) \oplus \delta_C(B)) = \delta(B) \setminus I_C(B). \end{aligned}$$

Therefore, in $yz_F(e)$ and $yz_C(e)$, z-values are summed up over the same set of blossoms in Ω. In other words, $yz_F(e) = yz_C(e)$ for each $e \in E$ and the claim follows

To prove that Property 3.9(3) implies Property 3.13(3), we argue by definition that the maturity of an f'-matching blossom implies the maturity of the corresponding f-edge-cover blossom. Equality is then implied by Lemma 3.4 and Lemma 3.6. Indeed, by how we define our f'-matching F and f-edge cover C, a vertex's surplus with respect to C and f is equal to a vertex's deficiency with respect to F and f'. Moreover,

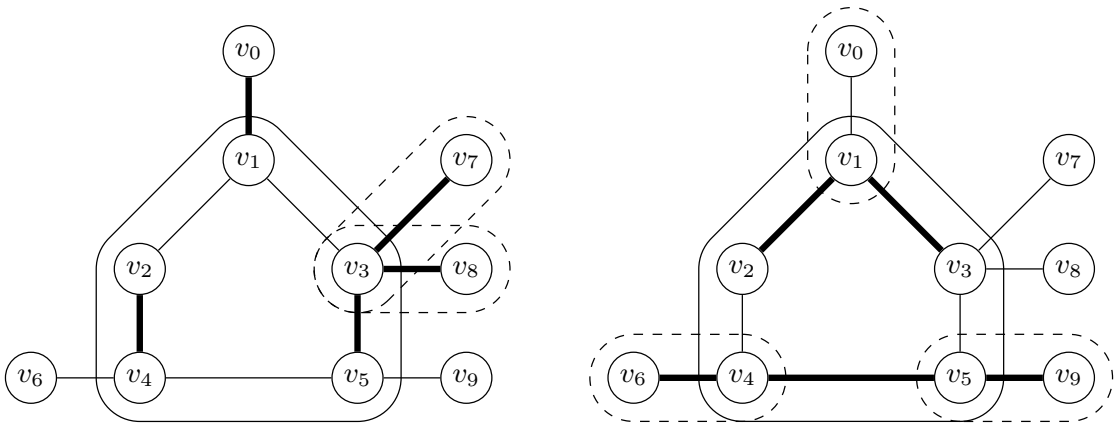


Figure 3.3: Illustration on relation between I-set of an f-matching and the I-set of its complementary f'-edge cover. Left: an f-matching and its blossom set. Right: Its complementary f-edge cover. Their I-sets are circled (dashed).

the blossom is heavy/light for f'-matching iff it is light/heavy for the corresponding f-edge cover. Since the base edge is defined to be the same, maturity of one blossom implies the other. This completes the proof. □

3.4 Approximation Algorithms for f-Matching and f-Edge Cover

In this section, we prove the main result by giving an approximation algorithm for computing $(1-\epsilon)$-approximate maximum weight f-matching. The crux of the result is an implementation of Edmonds' search with relaxed complementary slackness as the eligibility criterion. The notion of approximate complementary slackness was introduced by Gabow and Tarjan for both bipartite matching [74] and general matching [75]. Gabow gave an implementation of Edmonds' search with *exact* complementary slackness for the f-matching problem [71], which finds augmenting walks one at a time. The main contribution of this section is to adapt [71] to approximate complementary slackness to facilitate finding augmenting walks in batches.

To illustrate how this works, we will first give an approximation algorithm for f-matching in graphs with small edge weights. Let $w(\cdot)$ be a positive weight function $w : E \mapsto \{0, \ldots, W\}$. The algorithm computes a $(1-\epsilon)$-approximate maximum weight f-matching in $O(mW\epsilon^{-1})$ time, independent of f. We also show how to use scaling techniques to transform this algorithm to run in $O(m\epsilon^{-1}\log\epsilon^{-1})$ time, independent of W.

3.4.1 Approximation for Small Weights

The main procedure in our $O(mW\epsilon^{-1})$ time algorithm is a variation on Edmonds' search. In one iteration, Edmonds' search finds a set of augmenting walks using *eligible edges*, creates and dissolves blossoms, and performs *dual adjustments* on y and z while maintaining the following Invariant:

Invariant 3.20 (Approximate Complementary Slackness)**.** *Let $\delta > 0$ be some parameter such that $w(e)$ is a multiple of δ, for all $e \in E$:*

1. *Granularity. y-values are multiples of $\delta/2$ and z-values are multiples of δ.*
2. *Approximate Domination. For each unmatched edge and each blossom edge $e \in (E \setminus F) \cup (\bigcup_{B\in\Omega} E_B)$, $yz(e) \geq w(e) - \delta$.*
3. *Approximate Tightness. For each matched and each blossom edge $e \in F \cup (\bigcup_{B\in\Omega} E_B)$, $yz(e) \leq w(e)$.*
4. *Blossom Maturity. For each blossom $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$. Root blossoms in Ω have positive z-values.*
5. *Unsaturated Vertices. All unsaturated vertices have the same y-value; their y-values are strictly less than the y-values of other vertices.*

Notice that here we relax Property 3.9(4) to allow unsaturated vertices to have positive y-values. The purpose of Edmonds' search is to decrease the y-values for all unsaturated vertices while maintaining Invariant 3.20. Following [75, 55, 56], we define the following eligibility criterion:

Criterion 3.21. *An edge (u, v) is* eligible *if it satisfies one of the following:*

1. *$e \in E_B$ for some $B \in \Omega$.*
2. *$e \notin F$ and $yz(e) = w(e) - \delta$.*
3. *$e \in F$ and $yz(e) = w(e)$.*

A key property of this definition is that it is asymmetric for matched and unmatched edges that are not in any blossom. As a result, if we augment along an eligible augmenting walk P, all edges in P, except for those in contracted blossoms, will become ineligible; and its image in the contracted graph will become entirely ineligible.

We define $\mathcal{G}_{\text{elig}}$ to be the graph obtained from G by discarding all ineligible edges, and let $\widehat{\mathcal{G}_{\text{elig}}} = \mathcal{G}_{\text{elig}}/\Omega$ be obtained from $\mathcal{G}_{\text{elig}}$ by contracting all blossoms in Ω. For initialization, we set $F = \emptyset$, $y = W/2$, $z = 0$, $\Omega = \emptyset$. Edmonds' search repeatedly

1. *Augmentation.* Find a set of edge-disjoint augmenting walks $\widehat{\Psi}$ and a set of alternating cycles $\widehat{\mathcal{C}}$ in $\widehat{\mathcal{G}_{\text{elig}}}$, such that after removing their edges from $\widehat{\mathcal{G}_{\text{elig}}}$, $\widehat{\mathcal{G}_{\text{elig}}}$ does not contain any augmenting walk. Let Ψ and $\mathcal{C}$ be the preimages of $\widehat{\Psi}$ and $\widehat{\mathcal{C}}$ in $\mathcal{G}_{\text{elig}}$. Update $F \leftarrow F \oplus \left(\left(\bigcup_{P\in\Psi} P\right) \cup \left(\bigcup_{C\in\mathcal{C}} C\right)\right)$. After this step, the new $\widehat{\mathcal{G}_{\text{elig}}}$ contains no augmenting walk.
2. *Blossom Formation.* Find a maximal set Ω' of nested blossoms reachable from an unsaturated vertex/blossom in $\widehat{\mathcal{G}_{\text{elig}}}$. Update $\Omega \leftarrow \Omega \cup \Omega'$ and then update $\widehat{\mathcal{G}_{\text{elig}}}$ to be G/Ω. After this step, $\widehat{\mathcal{G}_{\text{elig}}}$ contains no blossom reachable from an unsaturated vertex/blossom.
3. *Dual Adjustment.* Let $\widehat{S}$ be the set of vertices from $\widehat{\mathcal{G}_{\text{elig}}}$ reachable from an unsaturated vertex via an eligible alternating walk. We classify vertices in $\widehat{S}$ into $\widehat{V_{in}}$, the set of inner vertices and $\widehat{V_{out}}$, the set of outer vertices.[5] Let V_{in} and V_{out} be the set of original vertices in V represented by $\widehat{V_{in}}$ and $\widehat{V_{out}}$. Adjust the y and z values as follows:

$$\begin{aligned} y(v) &\leftarrow y(v) - \delta/2, \text{ if } v \in V_{out} \\ y(v) &\leftarrow y(v) + \delta/2, \text{ if } v \in V_{in} \\ z(B) &\leftarrow z(B) + \delta, \text{ if } B \text{ is a root blossom in } \widehat{V}_{out} \\ z(B) &\leftarrow z(B) - \delta, \text{ if } B \text{ is a root blossom in } \widehat{V}_{in} \end{aligned}$$

   Here a root blossom is an inclusionwise maximal blossom in Ω.
4. *Blossom Dissolution.* After Dual Adjustment some root blossoms in Ω might have 0 z-values. Remove them from Ω until none exists. Update Ω and $\widehat{\mathcal{G}_{\text{elig}}}$.

Figure 3.4: A $(1-\epsilon)$-approximate f-matching algorithm for small integer weights.

executes the following steps: *Augmentation*, *Blossom Formation*, *Dual Adjustment*, and *Blossom Dissolution* until all unsaturated vertices have 0 y-values. See Figure 3.4.

Now we define what we mean by *reachable* vertices in Steps 1–3 of the algorithm, as well as the inner/outer labelling of nontrivial blossoms and singletons. This is analogous to the reachable/inner/outer vertices in Edmonds' Search for ordinary matching [55, 56], except that we cannot simply treat a contracted blossom like a single vertex. The corresponding definition for f-matching is given in Gabow [71, p. 46]. For completeness, we restate these definitions and further supplement them with the notion of *alternation*, which provides further insights for reachability.

We start by defining *alternation* which follows from Definition 3.7 of an augmenting walk. We say two distinct edges e, e' incident to a blossom/singleton B *alternate* if either B is a singleton and e and e' are of different types, or B is a nontrivial blossom and $|\eta(B) \cap \{e, e'\}| = 1$. An *alternating walk/cycle* in the contracted graph is a walk/cycle where every two consecutive edges alternates. An augmenting walk is an alternating walk with its terminal edges and terminal vertices satisfying the requirement specified in Definition 3.7.

$\widehat{S}$ is the set of blossoms and vertices in $\widehat{\mathcal{G}_{\text{elig}}}$ that are reachable from an unsaturated singleton or an unsaturated light blossom via an eligible alternating walk. It can be obtained by inductively constructing an alternating search tree rooted at an unsaturated singleton or an unsaturated light blossom. We label the root nodes *outer*. For a nonroot vertices v in $\widehat{S}$, let $\tau(v)$ be the edge in $\widehat{S}$ pointing to the parent of v. The inner/outer status of v is defined as follows:

Definition 3.22. *[71, p. 46] A vertex v is* outer *if one of the following is satisfied:*

1. *v is the root of a search tree.*
2. *v is a singleton and $\tau(v) \in F$.*
3. *v is a nontrivial blossom and $\{\tau(v)\} = \eta(v)$.*

Otherwise, one the the following holds and v is classified as inner*:*

1. *v is a singleton and $\tau(v) \in E \backslash F$.*
2. *v is a nontrivial blossom and $\{\tau(v)\} \neq \eta(v)$.*

An individual search tree in $\widehat{S}$, call it $\widehat{T}$, can be grown by repeatedly attaching a child v to its parent u using an edge (u, v) that is *eligible for u* in $\widehat{S}$; See Gabow [71, p. 46]. Let B_u denote the root blossom in Ω containing u. We say an edge $(u, v) \in E$

[5] In an actual implementation, the inner/outer labelling can be computed in the search in Blossom Formation step. The labelling continues to be valid after contracting a maximal set of blossoms.

is *eligible for* u if it is eligible according to Criterion 3.21 and one of the following is satisfied:

1. u is an outer singleton and $e \notin F$.
2. B_u is an outer blossom and $\{e\} \neq \eta(B_u)$.
3. u is an inner singleton and $e \in F$.
4. B_u is an inner blossom and $\{e\} = \eta(B_u)$.

Hence, $\widehat{S}$ consists of singletons and blossoms that are reachable from an unsaturated singleton or light blossom, via an eligible alternating path. We call such blossoms and singletons *reachable*, and all other singletons and blossoms *unreachable*. A vertex v from the original graph $\mathcal{G}_{\text{elig}}$ is reachable (unreachable) if B_v is reachable (unreachable) in $\widehat{\mathcal{G}_{\text{elig}}}$.[6]

In Edmonds' Search, primal and dual variables are initialized in a way that Property 3.9(1) (Approximate Domination) is always satisfied, and Property 3.9 (Approximate Tightness) is vacuous (as the f-matching is initially empty) but Property 3.9(4) (Unsaturated Vertices) is not. For this reason, there is a large gap between primal and dual objective, besides the error introduced by *approximate* tightness and domination, at the beginning of the algorithm. This gap is given by the following, assuming exact tightness and domination is satisfied (i.e. $\delta_1 = \delta_2 = 0$ in Property 3.9):

$$\begin{aligned} yz(V) - w(F) &= \sum_{v \in V} f(v)y(v) + \sum_{B \in \Omega} \left\lfloor \frac{f(B) + |I(B)|}{2} \right\rfloor z(B) + \sum_{e \in E} u(e) - \sum_{e \in F} w(e) \\ &= \sum_{v \in V} \text{def}(v)y(v). \end{aligned}$$

The goal of the algorithm can be seen as bridging the gap between the primal objective and dual objective while preserving all other complementary slackness properties. It can be achieve in two ways. Augmentations enlarge the f-matching by augmenting F along some augmenting walk P. This will reduce the total deficiency on the vertex set V. Dual Adjustments change the dual variables in a way that decreases the y-value on unsaturated vertices while maintaining other approximate complementary slackness conditions. In this algorithm, the progress of Edmonds' Search is measured by the latter, i.e., the overall reduction in y-values of unsaturated vertices.

[6]Of course, if B_v is inner and reachable in $\widehat{G}$, this only implies that $\beta(B_v)$ is reachable from an unsaturated vertex in G; other vertices in B_v may not be reachable in G.

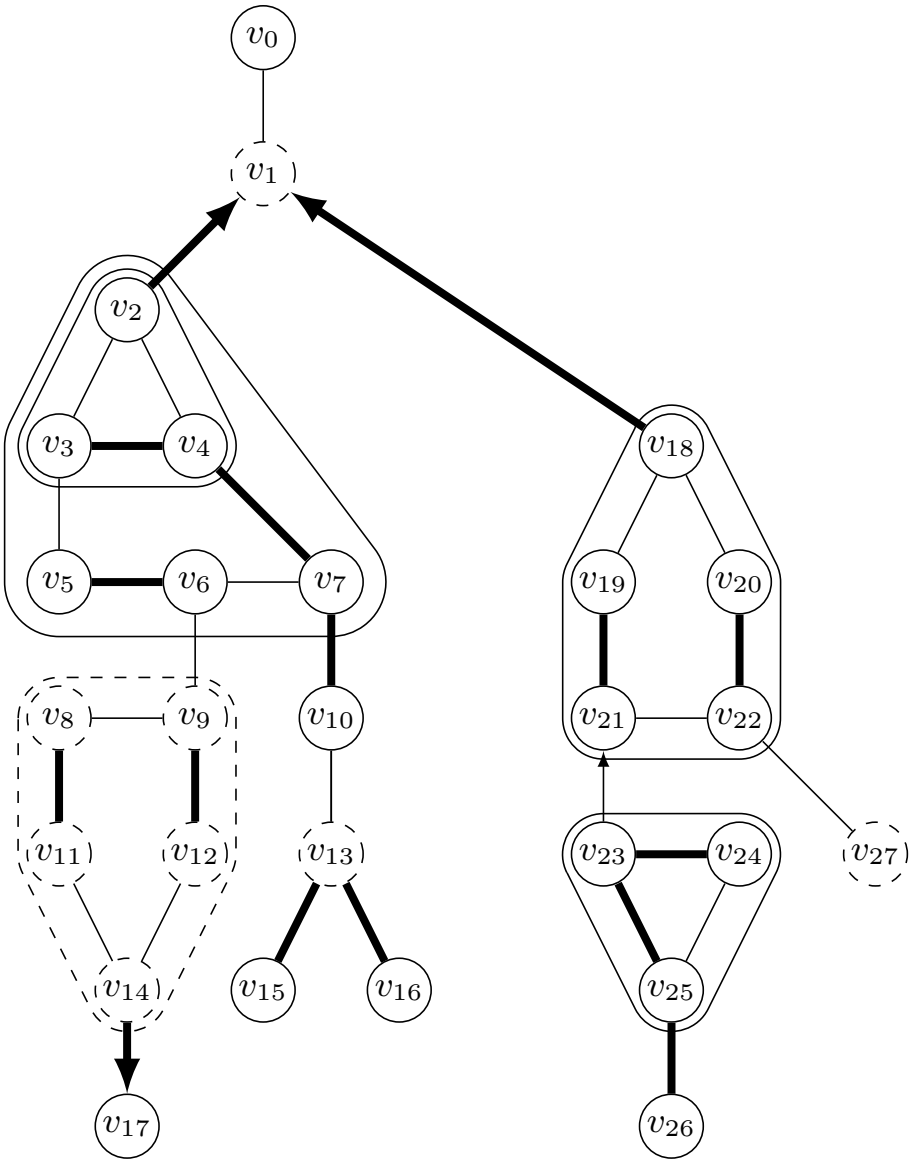


Figure 3.5: An example of an eligible alternating search tree. Outer blossoms and singletons are labeled using solid boundaries while inner blossoms and singletons have dashed boundaries.

The correctness of our algorithm reduces to showing that *Augmentation*, *Blossom Formation*, *Blossom Dissolution*, and *Dual Adjustment* all preserve Invariant 3.20.

Lemma 3.23. *The Augmentation step, Blossom Formation step and Blossom Dissolution step preserve Invariant 3.20.*

Proof. We first show that the identity of $I(B)$ is invariant under an augmentation; in particular, augmenting along an augmenting walk that intersects B does not change $I(B)$. As a result, the function $yz(\cdot)$ is invariant under augmentation. This is a restatement of Lemma 5.3 in [71], for completeness, we restate the proof.

We use $I(B), \eta(B)$ and $I'(B), \eta'(B)$ to denote the I-set and base edge of B before and after the augmentation. By Definition 3.7 (augmenting walks), if P intersects B, then

$$\delta_P(B) = \eta(B) \cup \eta'(B) = \eta(B) \oplus \eta'(B).$$

Let F and F' be the f-matching before and after augmentation. We have

$$\delta_{F'}(B) = \delta_F(B) \oplus \delta_P(B)$$

Combining both equations, we have

$$\delta_{F'}(B) = \delta_F(B) \oplus (\eta(B) \oplus \eta'(B))$$

Hence

$$I'(B) = \delta_{F'}(B) \oplus \eta'(B) = \delta_F(B) \oplus \eta(B) = I(B).$$

By Invariant 3.20, any blossom edge $e \in \bigcup_{B \in \Omega} E_B$ satisfies both approximate domination as well as approximate tightness, so it continues to satisfy these Invariants after augmentation. For any eligible edge not in E_B for any $B \in \Omega$, by Criterion 3.21, if e is matched, $yz(e) = w(e) - \delta$, thus after the Augmentation step its duals satisfy approximate domination. If e is unmatched, $yz(e) = w(e)$, so its duals satisfy approximate tightness after the Augmentation step.

Augmentation also preserves the maturity of blossoms. For any vertex v in a nonterminal blossom B, $\deg_F(v) = \deg_{F'}(v) = f(v)$, so maturity is naturally preserved. If B is a terminal blossom, we have $\deg_F(v) = f(v) - 1$ for $v = \beta(B)$ and $\deg_F(v) = f(v)$ for all $v \neq \beta(B)$. Moreover, after augmentation B always has a base edge $\eta(B) = \delta_P(B)$. Therefore, B is also mature after augmentation.

All the newly formed blossoms in this step must be mature and have 0 z-values, so the value of the yz function is unchanged and all the invariants are preserved.

For blossom dissolution step, discarding blossoms with zero z-values preserves the value of the yz function and hence preserves the invariants. □

The crux of the proof is to show that Dual Adjustment also preserves Invariant 3.20, in particular approximate domination and approximate tightness. Before proving the correctness of Dual Adjustment, we first prove the following parity lemma, which was first used in [75]; we generalize it to f-matching:

Lemma 3.24 (Parity)**.** *Let $\widehat{S}$ be the search forest defined as above. Let S be the preimage of $\widehat{S}$ in G. The y-value of every vertex in S has the same parity, as a multiple of $\delta/2$.*

Proof. The claim clearly holds after initialization as all vertices have the same y-values. Now notice that every *eligible edges* $e = (u, v)$ that straddles two distinct singletons or nontrivial blossom must have its yz-value being $w(e)$ or $w(e) - \delta$. Since $w(e)$ is by assumption an integral multiple of δ, $yz(e)$ is also a multiple of δ. Because z-values are always multiples of δ, $y(u)$ and $y(v)$ must both be odd or even as a multiple of $\delta/2$.

Therefore it suffices to show that every vertex in a blossom $B \in \Omega$ has the same parity.

To prove this, we only need to show that the Blossom Formation step only groups vertices with the same parity together. This is because new blossoms B are formed when we encounter a cycle of nontrivial blossoms and singletons $C_B = \langle B_0, e_0, B_1, e_1, \dots, B_{l-1}, e_{l-1} \rangle$ whose edges are eligible. Therefore the endpoints of those edges share the same parity. Hence by induction, all vertices in B also share the same parity. The Dual Adjustment step also preserves this property as vertices in a blossom will have the same inner/outer classification and thus have their y-values all incremented or decremented by $\delta/2$. □

The following theorem is a generalization of Lemma 5.8 in [71] to approximate complementary slackness. The proof follows from the same framework but has a slightly more complicated case analysis.

Lemma 3.25. *Dual Adjustment and Blossom Dissolution preserves Invariant 3.20.*

Proof. We focus on part 2 (Approximate Domination) and part 3 (Tightness) of Property 3.9. Part 1 (Granularity) is naturally preserved since we are adjusting y-values by $\delta/2$ and z-values by δ. Part 5 (Unsaturated vertices duals) is also preserved because unsaturated vertices are labelled as outer and their dual is adjusted by the same

amount. Maturity of blossoms is not affected by Dual Adjustment. Although after dual adjustment, some (inner) root blossoms might have 0 z-values, such blossoms are removed in Blossom Dissolution step so part 4 for Invariant 3.20 is restored at the end of the iteration.

Similar to ordinary matching, preservation of approximate domination and tightness can be argued using a case analysis on vertices and blossoms dual. Notice that there are more cases to consider in f-matching compared to ordinary matching. Different cases can be generated for an edge (u, v) by considering the inner/outer classification of both endpoints, whether (u, v) is matched, whether (u, v) is the base edge for its respective endpoints, if they are in blossoms, and whether (u, v) is eligible. In the following analysis, we follow the framework in Lemma 5.8 [71] to narrow down the number of meaningful cases to just 8. Notice that Lemma 5.8 [71] can be seen as a version of this lemma for exact complementary slackness. Although one can expect the same conclusion to hold, the proof still differs in details.

We consider an edge $e = (u, v)$. If u and v are both unreachable, or both in the same root blossom, $yz(u, v)$ clearly remains unchanged after Dual Adjustment.

Therefore we can assume $B_u \neq B_v$ and at least one of them, say B_u, is reachable. Every reachable endpoint will contribute a change of $\pm\delta/2$ to $yz(u, v)$. This is the adjustment of $y(u)$, plus the adjustment of $z(B_u)$ if $e \in I(B_u)$. Define $\Delta(u)$ to be the net change of the quantity $y(u) + \sum_{e \in I(B_u)} z(B_u)$. By definition of Dual Adjustment, we have the following scenarios:

- $\Delta(u) = +\delta/2$: This occurs if u is an inner singleton, or B_u is an outer blossom with $e \in I(B_u)$, or an inner blossom with $e \notin I(B_u)$.
- $\Delta(u) = -\delta/2$: This occurs if u is an outer singleton, or B_u is an inner blossom with $e \in I(B_u)$, or an outer blossom with $e \notin I(B_u)$.

Then we consider the effect of a Dual Adjustment on the edge $e = (u, v)$. First we consider the case when exactly one of B_u and B_v, say B_u, is in $\widehat{S}$. In this case only u will introduce a change on $yz(u, v)$:

Case *1:* u is an inner singleton: Here $\Delta(u) = +\delta/2$. In this case approximate domination is preserved, so we only need to worry about approximate tightness and hence assume $e \in F$. Since B_v is not in $\widehat{S}$, e cannot be eligible for B_u or B_v would have been included in $\widehat{S}$ as a child of B_u. Because $e \in F$, e cannot be eligible. Hence $yz(e) < w(e)$. By Granularity, $yz(e) \leq w(e) - \delta/2$. Therefore we have $yz(e) \leq w(e)$ after the Dual Adjustment.

Case *2:* u is an outer singleton: Here $\Delta(u) = -\delta/2$. In this case tightness is preserved and we only need to worry about approximate dominination when $e \notin F$. Similar to Case 1, e must be ineligible and $yz(e) \geq w(e) - \delta/2$. After Dual Adjustment we have $yz(e) \geq w(e) - \delta$.

Case *3:* B_u is an inner blossom: We divide the cases according to whether e is matched or not.

Subcase 3.1: $e \in F$. If $e \notin \eta(B_u)$, then $e \in I(B_u)$ and $\Delta(u) = -\delta/2$. In this case tightness is preserved. If $e \in \eta(B_u)$, then $e \notin I(B_u)$ and $\Delta(u) = +\delta/2$. But e cannot be eligible since otherwise B_v would be in the search tree, so we have $yz(e) \leq w(e) - \delta/2$ and $yz(e) \leq w(e)$ after Dual Adjustment.

Subcase 3.2: $e \notin F$. This is basically symmetric to Subcase 3.1. If $e \in \eta(B_u)$, then $e \in I(B_u)$ and $\Delta(u) = -\delta/2$. But e cannot be eligible therefore $yz(e) \geq w(e) - \delta/2$, and $yz(e) \geq w(e) - \delta$ after Dual Adjustment. If $e \notin \eta(B_u)$, then $e \notin I(B_u)$ and $\Delta(u) = +\delta/2$, so approximate Domination is preserved.

Case *4:* B_u is an outer blossom:

Subcase 4.1: $e \in F$. If $e \in \eta(B_u)$, then B_v must be the parent of B_u in the search tree, contradicting the fact that $B_v \notin \widehat{S}$. Thus $e \notin \eta(B_u)$, so $e \in I(B_u)$ and $\Delta(u) = +\delta/2$. Since B_v is not reachable, e cannot be eligible, so $yz(u, v) \leq w(e) - \delta/2$ before Dual Adjustment and $yz(u, v) \leq w(e)$ afterward.

Subcase 4.2: $e \notin F$. Similarly, $e \notin \eta(B_u)$, so $e \notin I(B_u)$ and $\Delta(u) = -\delta/2$. Similarly B_v is not reachable so e cannot be eligible. Therefore we have $yz(u, v) \geq w(e) - \delta/2$ and $yz(u, v) \geq w(e) - \delta$ after Dual Adjustment.

This completes the case when exactly one of e's endpoints is reachable. The following part will complete the argument for when both endpoints are reachable. We argue that three scenarios can happen: either $\Delta(u)$ and $\Delta(v)$ are of opposite signs and cancel each other out, or $\Delta(u)$ and $\Delta(v)$ are of the same sign and the sign aligns with the property we wish to keep, or if neither case holds, we use Lemma 3.24 (Parity) to argue that there is enough room for dual adjustment not to violate approximate domination or tightness.

We first examine tree edges in $\widehat{S}$. In this case we assume B_u is the parent of B_v and e is the parent edge of B_v. Hence e must be eligible for B_u. We argue by the sign of $\Delta(u)$.

Case *5:* If e is a tree edge and $\Delta(u) = +\delta/2$:

There are three cases here: u is an inner singleton, B_u is an outer blossom with $e \in I(B_u)$, or B_u is an inner blossom with $e \notin I(B_u)$. We first observe that in all three cases, $e \in F$. This is straightforward when u is an inner singleton. If B_u is an outer blossom with $e \in I(B_u)$, we know that since B_u is outer, $e \notin \eta(B_u)$, so therefore $e \in F$. If B_u is an inner singleton with $e \notin I(B_u)$, since B_u is inner, $e \in \eta(B_u)$, so combined with the fact that $e \notin I(B_u)$ we have $e \in F$.

Notice that since B_u is the parent of B_v, and $e \in F$, v can be an outer singleton, or B_v is an outer blossom with $e \in \eta(B_v)$, or B_v is an inner blossom with $e \notin \eta(B_v)$. In the second case $e \notin I(B_v)$ and in the third case $e \in I(B_v)$. In all three cases we have $\Delta(v) = -\delta/2$, and $yz(e)$ remains unchanged.

Case *6:* If e is a tree edge and $\Delta(u) = -\delta/2$: Case 6 is symmetric to Case 5. B_u can either be an outer singleton, an inner blossom with $e \in I(B_u)$, or an outer blossom with $e \notin I(B_u)$. In all cases, the fact that e must be eligible for B_u implies $e \notin F$, and B_v can only be an inner singleton, an outer blossom with $e \in I(B_u)$, or an inner blossom with $e \notin I(B_v)$. Hence we have $\Delta(v) = +\delta/2$ so $yz(e)$ remains unchanged.

Now suppose B_u and B_v are both in $\widehat{S}$ but (u, v) is not a tree edge. We still break the cases according to the sign of $\Delta(u)$ and $\Delta(v)$. Here we only need to consider when $\Delta(u) = \Delta(v)$, since otherwise they cancel each other and $yz(e)$ remains constant.

Case *7:* If e is a tree edge and $\Delta(u) = \Delta(v) = \delta/2$. In this case $yz(e)$ is incremented by δ. Therefore we only need to worry about tightness when $e \in F$. Notice that B_u can only be an inner singleton, an outer blossom with $e \in I(B_u)$ or an inner blossom with $e \notin I(B_u)$. When B_u is an outer blossom, $e \notin \eta(B_u)$. When B_u is an inner blossom, since $e \in F$ and $e \notin I(B_u)$, $e \in \eta(B_u)$. The same holds for the other endpoint B_v.

It is easy to verify that in all cases, e is eligible for B_u (or B_v) if and only if e is eligible. But notice that after Augmentation and Blossom Formation steps, there is no augmenting walk or reachable blossom in $\widehat{\mathcal{G}_{\text{elig}}}$, i.e., there cannot be an edge (u, v) that is eligible for both endpoints B_u and B_v since otherwise one can find an augmenting walk or a new reachable blossom. Thus e is ineligible and $yz(e) < w(e)$. But by Invariant 3.20(1) (Granularity) and Lemma 3.24 (Parity), both $w(e)$ and $yz(e)$ must be multiples of δ. Therefore we have $yz(e) \leq w(e) - \delta$. This implies $yz(e) \leq w(e)$ after Dual Adjustment.

Case *8:* If e is a tree edge and $\Delta(u) = \Delta(v) = -\delta/2$. Here $yz(e)$ is decremented by δ. Similar to the case above, we can assume $e \notin F$ and only focus on approximate

domination. B_u can be an outer singleton, inner blossom with $e \in I(B_u)$, or outer blossom with $e \notin I(B_u)$. Since $e \notin F$, $e \in I(B_u)$ if and only if $e \in \eta(B_u)$. Therefore if e is eligible, e must be eligible for both B_u and B_v. But similar to Case 7, e being eligible for both endpoints will lead to the discovery of an additional blossom or augmenting walk in $\mathcal{G}_{\text{elig}}$, which is impossible after Augmentation and Blossom Formation. Therefore we conclude in this case e is ineligible and $yz(e) > w(e) - \delta$. By Lemma 3.24 (Parity), we have $yz(e) \geq w(e)$ before Dual Adjustment, so approximate domination still holds after Dual Adjustment. □

Theorem 3.26. *A $(1-\epsilon)$-approximate f-matching can be computed in $O(Wm\epsilon^{-1})$ time.*

Proof. We initialize the f-matching to be $\emptyset$ and $y(v) = W/2$ for all v. Set $\delta = 1/\lceil \epsilon^{-1} \rceil \leq \epsilon$. Since each iteration decreases y-values by $\delta/2$, y-values of unsaturated vertices takes $(W/2)/(\delta/2) = O(W\epsilon^{-1})$ iterations to reach 0, thereby satisfying Property 3.9 with $\delta_1 = \delta, \delta_2 = 0$. By invoking Lemma 3.10, with F^* being the optimum f-matching, we have

$$w(F) \geq w(F^*) - |F^*|\delta \geq w(F^*) - w(F^*)\delta \geq (1-\epsilon)w(F^*).$$

For the running time, each iteration of Augmentation, Blossom Formation, Dual Adjustment, and Blossom Dissolution can be implemented in linear time. We defer the detailed implementation to Section 3.5. There are a total of $W/\delta = O(W\epsilon^{-1})$ iterations, so the running time is $O(Wm\epsilon^{-1})$. □

As a result of Lemma 3.19 and Lemma 3.15, we also obtain the following result:

Corollary 3.27. *A $(1+\epsilon)$-approximate f-edge cover can be computed in $O(Wm\epsilon^{-1})$ time.*

Proof. Given a weighted graph G and degree constraint function f, let $f' = \deg - f$ be the complement of f. With some paramter δ we run the algorithm from Theorem 3.26 to find an f'-matching F' that satisfies Property 3.9 with parameters $(\delta, 0)$. By Lemma 3.19, its complement $F = E \setminus F'$ satisfies Property 3.13 with parameter $(0, \delta)$. By Lemma 3.15, we have

$$w(F) - \delta|F| \leq w(F^*)$$
$$(1-\delta)w(F) \leq w(F^*)$$

Then we can choose a $\delta = \Theta(\epsilon)$ to guarantee that we get an $(1+\epsilon)$-approximate minimum weight f-edge cover. □

Also notice that when $W = O(1)$ is constant, Theorem 3.26 and Corollary 3.27 are the fastest known approximation algorithms for these problems.

3.4.2 A Scaling Algorithm for General Weights

In this section, we can modify the $O(Wm\epsilon^{-1})$ weighted f-matching algorithm to work on graphs with general real weights. The modification is based on the scaling framework in [55]. If the weights are arbitrary reals, we can round them to integers in $[W]$, $W = \text{poly}(n)$, with negligible loss in accuracy. Thus we can assume without loss of generality that all weights are $O(\log n)$-bit integers. The idea is to divide the algorithm into into $L = \log W + 1$ scales that execute Edmonds' search with exponentially diminishing δ. The goal of each scale is to use $O(\epsilon^{-1})$ Edmonds' searches to halve the y-values of all unsaturated vertices while maintaining a more relaxed version of approximate complementary slackness. By manipulating the weight function, approximate domination, which is weak at the beginning, is strengthened over scales, while approximate tightness is weakened in exchange. Assume without loss of generality that $W > 1$ and $\epsilon < 1$ are powers of two. We define $\delta_i, 0 \leq i \leq L$ be the error parameter for each scale, where $\delta_0 = \epsilon W$ and $\delta_i = \delta_{i-1}/2$ for $0 < i \leq L$. Each scale works with a new weight function w_i which is the old weight function rounded down to the nearest multiple of δ_i, i.e, $w_i(e) = \delta_i \lfloor w(e)/\delta_i \rfloor$. In the last scale $W_L = w$. We maintain a scaled version of Invariant 3.20 at each scale:

Invariant 3.28 (Scaled approximate complementary slackness with positive unsaturated vertices)**.** *At scale $i = 0, 1, \ldots, L = \log W$, we maintain the f-matching F, blossoms Ω, and duals y, z to satisfy the following invariant:*

1. *Granularity. All y-values are multiples of $\delta_i/2$, and z-values are multiples of δ_i.*
2. *Approximate Domination. For each $e \notin F$ or $e \in E_B$ for some $B \in \Omega$, $yz(e) \geq w_i(e) - \delta_i$.*
3. *Approximate Tightness. For each $e \in F \cup (\bigcup_{B\in\Omega} E_B)$, let $j_e \leq i$ be the index of last scale that e joined the set $F \cup \bigcup_{B\in\Omega} E_B$. We have $yz(e) \leq w_i(e) + 2\delta_{j_e} - 2\delta_i$.*
4. *Mature Blossoms. For each blossom $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$.*
5. *Unsaturated Vertices' Duals. The y-values of all unsaturated vertices are the same and less than the y-values of other vertices.*

Based on Invariant 3.28, Edmonds' search will use the following Eligibility criterion:

Criterion 3.29. *At scale i, an edge $e \in E$ is eligible if one of the following holds*

1. *$e \in E_B$ for some $B \in \Omega$.*
2. *$e \notin F$ and $yz(e) = w_i(e) - \delta_i$.*
3. *$e \in F$ and $yz(e) - w_i(e)$ is a nonnegative integer multiple of δ_i.*

This is similar to Criterion 3.21 except for we have a relaxed criterion for when $e \in F$. This relaxation is due to the fact that tightness is weakened at termination of each scale, and the eligibility criterion is then relaxed to accommodate it. We argue below that this relaxation does not affect the correctness of Edmonds' Search.

Before the start of scale 0, the algorithm initializes F, Ω, y, z similar to the algorithm for small edge weights: $y(u) \leftarrow W/2$, $\Omega \leftarrow \emptyset$, $F \leftarrow \emptyset$. At scale i, the duals of unsaturated vertices start at $W/2^{i+1}$. We execute $(W/2^{i+2})/(\delta_i/2) = O(\epsilon^{-1})$ iterations of Edmonds' search with parameter δ_i, using Criterion 3.29 of eligibility. The scale terminates when the y-values of unsaturated vertices are reduced to $W/2^{i+2}$, or in the last iteration, as they reach 0.

Notice that although the invariant and the eligibility criterion are changed, the fact that Edmonds' search preserves the complementary slackness invariant still holds. The proof of Lemma 3.25 goes through, as long as the definition of eligibility guarantees the following parity property:

Lemma 3.30. *At any point of scale i, let $\overline{S}$ be the set of vertices in $\mathcal{G}_{\text{elig}}$ reachable from an unsaturated vertex using eligible edges. The y-value of any vertex $v \in V$ with $B_v \in \overline{S}$ has the same parity as a multiple of $\delta_i/2$.*

We omit the proof of Lemma 3.30. The details are similar to Lemma 3.25, using Criterion 3.29 in lieu of Criterion 3.21.

Now we sketch why Criterion 3.29 ensures Invariant 3.28, in particular, how it ensures approximate domination and approximate tightness. We will not prove it formally as the details are very similar to Lemma 3.25 and Lemma 3.23.

Observe that primal and dual variables initially satisfy Invariant 3.28, in particular parts 2 and 3. This is because all edges have yz-values equal to W, and no edge is in $M \bigcup_{B \in \Omega} E_B$.

Notice that dual adjustment never changes the yz-values of edges inside any blossom $B \in \Omega$, while it will have the following effect on edge e if its endpoints lie in different blossoms.

1. If $e \notin F$ and is ineligible, $yz(e)$ might decrease but will never drop below the threshold for eligibility, i.e., it will not drop below $w_i(e) - \delta_i$.
2. If $e \notin F$ and is eligible, $yz(e)$ will never decrease.
3. If $e \in F$ and is ineligible, $yz(e)$ might increase but will never exceed the threshold for eligibility, i.e., it will not raise above $w_i(e) + 2\delta_{j_e} - 2\delta_i$.
4. If $e \in F$ and is eligible, $yz(e)$ will never increase.

In other words, with the proper definition of Eligibility, Dual Adjustment will not destroy approximate domination and approximate tightness. Therefore Edmonds' search within scale i will preserve Invariant 3.28.

We also need to manipulate the duals between different scales to ensure Invariant 3.28. Formally, after completion of scale i, we increment all the y-values by δ_{i+1}, i.e., if yz' and yz are the function before and after dual adjustment, $yz(e) = yz'(e) + 2\delta_{i+1}$. No change is made to F, Ω and z. This will ensure both approximate domination and approximate tightness hold at scale $i+1$. At the previous scale we have approximate domination $yz(e) \geq w_i(e) - \delta_i$. The weights at scale i and $i+1$ satisfy $w_{i+1}(e) \leq w_i(e) + \delta_{i+1}$. Thus, after dual adjustment,

$$\begin{aligned} yz(e) &= yz'(e) + 2\delta_{i+1} \\ &\geq w_i(e) - \delta_i + 2\delta_{i+1} \\ &\geq w_{i+1}(e) - \delta_{i+1} - \delta_i + 2\delta_{i+1} \\ &= w_{i+1}(e) - \delta_{i+1} \end{aligned}$$

For approximate tightness, we have

$$yz(e) - w_{i+1}(e) \leq yz(e) - w_i(e) \leq 2\delta_{j_e} - 2\delta_i + 2\delta_{i+1} = 2\delta_{j_e} - 2\delta_{i+1},$$

since $\delta_{i+1} = \delta_i/2$.

This step is the main motivation for the definition of Invariant 3.28 (3), as approximate tightness is gradually relaxed in this step. The algorithm terminates when the y-values of all unsaturated vertices reach 0. It terminates with an f-matching F and its corresponding duals y, z and Ω satisfying the following property:

Property 3.31 (Final Complementary Slackness)**.**

1. *Approximate Domination. For all $e \notin F$ or $e \in E_B$ for any $B \in \Omega$, $yz(e) \geq w(e) - \delta_L$.*

2. *Approximate Tightness. For all* $e \in F \cup (\bigcup_{B \in \Omega} E_B)$*, let* j_e *be the index of the last scale that* e *joined* $F \cup (\bigcup_{B \in \Omega} E_B)$*. We have* $yz(e) \leq w(e) + 2\delta_{j_e}$*.*
3. *Blossom Maturity. For all blossoms* $B \in \Omega$*,* $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(V)+|I(B)|}{2} \right\rfloor$*).*
4. *Unsaturated Vertices' Duals. The* y*-values of all unsaturated vertices are* 0*.*

This implies approximate domination and approximate tightness are satisfied within some factor $1 \pm O(\epsilon)$. For approximate domination this is easy to see since $w(e) \geq 1$ and $\delta_L = \epsilon/2$, thus $yz(e) \geq (1 - \epsilon/2)w(e)$ if $e \notin F$. For approximate tightness, we can lower bound the weight of e if e last entered F or a blossom at scale $j = j_e$. Throughout scale j, the y-values are at least $W/2^{j+2}$, so $w(e) \geq w_j(e) \geq 2(W/2^{j+2}) - \delta_j$. Since $\delta_j = \epsilon W/2^j$, $yz(e) \leq w(e) + 2\delta_j \leq (1 + 4\epsilon)w(e)$ when $e \in F$.

This implies we have $O(\epsilon)$ multiplicative error for both approximate domination and approximate tightness. Together with the Lemma 3.12, we can show F is a $(1 - \epsilon)$-approximate maximum weight f-matching.

The running time of the algorithm is $O(m\epsilon^{-1} \log W)$ because there are $\log W + 1$ scales, and each scale consists of $O(\epsilon^{-1})$ iterations of Edmonds' search, which can be implemented in linear time.

Theorem 3.32. *A* $(1 - \epsilon)$*-approximate maximum weight* f*-matching can be computed in* $O(m\epsilon^{-1} \log W)$ *time.*

Corollary 3.33. *A* $(1 + \epsilon)$*-approximate minimum weight* f*-edge cover can be computed in* $O(m\epsilon^{-1} \log W)$ *time.*

3.4.3 A Linear Time Algorithm

We also point out that by applying techniques in [55, §3.2], the algorithm can be modified to run in time independent of W. The main idea is to force the algorithm to ignore an edge e for all but $O(\log \epsilon^{-1})$ scales. First, we index edges by the first scale that it can ever become eligible. Since at scale i, y-values can drop at most to $W/2^{i+1}$, any edge with weight below $W/2^i$ cannot be eligible at scale i. Let $\mu_i = W/2^i$ and scale(e) be the unique i such that $w(e) \in [\mu_i, \mu_{i-1})$. Notice that we can ignore e in any scale $j <$ scale(e). Moreover, we will also forcibly ignore e at scale $j >$ scale$(e) + \lambda$ where $\lambda = \log \epsilon^{-1} + O(1)$. Ignoring an otherwise eligible edge might cause violations of approximate tightness and approximate domination. However, since the y-values of free vertices are $O(\epsilon w(e))$ at this point, this violation will only amount to $O(\epsilon w(e))$.

To see this, notice that μ_i is also an upper bound to the amount of change to $yz(e)$ caused by all Dual Adjustment *after* scale i. Hence, after scale $\text{scale}(e)+\lambda$, the total amount of violation to approximate tightness and approximate domination on e can be bounded by $\mu_{\text{scale}(e)+\lambda} = O(\epsilon)\mu_{\text{scale}(e)} = O(\epsilon)w(e)$, which guarantees we still get a $(1-O(\epsilon))$-approximate solution.

Therefore, every edge takes part in at most $\log \epsilon^{-1} + O(1)$ scales, with $O(\epsilon^{-1})$ cost per scale. The total running time is $O(m\epsilon^{-1}\log\epsilon^{-1})$. We are omitting the full proof of Theorem 3.34.

Theorem 3.34. *A $(1-\epsilon)$-approximate maximum weight f-matching and a $(1+\epsilon)$-approximate minimum weight f-edge cover can be computed in $O(m\epsilon^{-1}\log\epsilon^{-1})$ time, independent of the weight function.*

3.5 A Linear Time Augmenting Walk Algorithm

In this section, we show how to implement the augmentation and blossom formation steps in linear time. The goal of the augmentation step is to find a set of augmenting walks *and* alternating cycles in the contracted eligible subgraph, such that after the removal of these cycles and walks, the subgraph no longer contains any augmenting walks. In the blossom formation step, we are given a contracted graph without any augmenting walks. The goal is to find a maximal set of reachable and contractable blossoms, i.e., a set of blossoms whose contraction will leave the graph without any reachable and contractable blossoms.

We formalize this problem, called *Disjoint Paths and Blossoms Problem*, as follows:

Definition 3.35. *In the* Disjoint Path and Blossoms Problem*, we are given a graph $G=(V,E)$, where V is partitioned into two sets V_s, V_b, an f-matching M, and a partial function $\eta : V_b \mapsto E$ such that $\eta(v) \in \delta(v)$ if $\eta(v)$ exists. Here $v \in V_b$ represents a contracted blossom and $\eta(v)$ the incident base edge, if any. The goal is to find a set of alternating cycles $\mathcal{C}$, a set of augmenting walks Ψ where all cycles and walks are mutually edge disjoint, such that after removing all edges in $\mathcal{C}$ and Ψ, the remaining graph G does not contain any augmenting walks. Moreover, we also output a laminar set of nested blossoms Ω' on V, such that after contracting blossoms in Ω', the new contracted graph no longer contains any reachable and contractable blossom.*

There are several subtleties in this definition. G here is used as a contracted graph obtained by contracting a set of nested blossoms Ω from an underlying graph. There-

fore, augmenting walks and alternating cycles are defined according to Definition 3.7 and the definition of *alternation* from Section 3.4.1, by treating $\eta(v)$ as v's base edge when v represents a nontrivial blossom. As a result, an alternating cycle in G might not have even length in G and an augmenting walk might not have odd length in G. It is guaranteed, by Lemma 3.2, that the pre-images of these walks and cycles in the underlying graph are odd and even, respectively. Moreover, it is *not* guaranteed that no augmenting walk exists in the underlying graph after removing the image of Ψ and $\mathcal{C}$ in it.[7] However, it is sufficient since in the proof of Lemma 3.25, we only use the fact that the *contracted graph* does not contain any augmenting walks.

This problem is noticeably different from the problem solved in [75, §8] for 1-matching. Instead of looking for a *maximal* set of *vertex disjoint* augmenting paths, we look for a set of *edge disjoint* augmenting walks Ψ in conjunction with a set of *alternating cycles* $\mathcal{C}$ whose removal removes all augmenting walks from G.

Both algorithms try to search for a set of augmenting paths/walks by building an alternating structure S (not necessarily tree) whose topology is defined in Section 3.4.1. However, in 1-matching, the search structure branches only at outer singletons and blossoms, while in f-matching, it also branches at inner singletons. As a result, when a search process arrives at a vertex v, it also carries an inner/outer tag to remind the algorithm whether it is looking for an unmatched/non-η edge, or a matched/η edge to continue extending the structure.

A key difference between 1-matching and f-matching is that augmenting walks can be non-simple, i.e., they may contain an alternating cycle as a subwalk. Consequently, when the search process reaches an outer (inner) singleton u, it can potentially find, through an unmatched (matched) edge an inner (outer) singleton v that has already been visited before in the same search, and proceed to discover an augmenting walk. This phenomenon is illustrated in Figure 3.6. If the algorithm intends to discover $(v_0, v_1, v_2, v_3, v_4, v_1, v_5, v_6)$ as an augmenting walk, it will reach v_1 with inner tag twice; first from v_0, then from v_4. Notice that in ordinary matching, edge (v_1, v_5) cannot exist and edge (v_4, v_1) is ignored as it provides no useful information regarding whether v_1 is an inner/outer vertex.

One might propose to ignore and discard the edge (v_4, v_1) and return the simple path (v_0, v_1, v_5, v_6). However, edges like (v_4, v_1) cannot simply be discarded from future searches as they might participate in other augmenting walks, say $(v_{10}, v_9, v_4, v_1, v_8, v_7)$

[7] This is because multiple augmenting walks in the underlying graph can intersect a single blossom in Ω before we contract the blossom, while after contracting a blossom, any augmenting walk or alternating cycle going through the blossom will forbid the other walks and cycle to use the same blossom again (as it must go through the base edge).

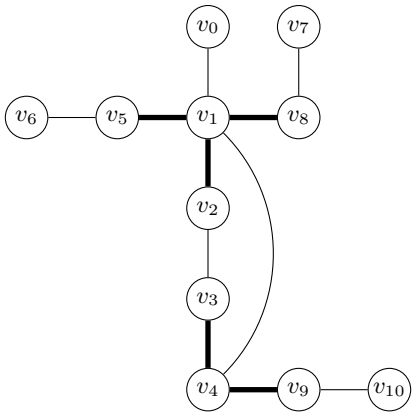


Figure 3.6: Example of a self-intersecting search structure and nonsimple augmenting walk. Here v_0 is the root of the search structure and $\{v_0, v_2, v_4, v_5\}$ is the set of outer vertices and $\{v_1, v_3\}$ is the set of inner vertices. The search begins with v_0 and proceed to v_1, v_2, v_3, v_4 in order. The procedure then scan the edge (v_4, v_1) and because it connect an outer vertex to an inner vertex that is already visited, it might ignore the edge and backtrack to v_1 and return the augmenting walk $\langle v_0, v_1, v_5, v_6\rangle$. However, although (v_1, v_4) is scanned and ignored, it cannot be discarded from future search as another augmenting walk, such as the dashed walk $\langle v_7, v_8, v_1, v_4, v_9, v_{10}\rangle$ might make use of the edge (v_1, v_4) and Ψ will not be maximal if (v_1, v_4) participating in some augmenting walks.

that is edge disjoint from (v_0, v_1, v_5, v_6). To achieve a linear running time, it is essential that edges of this type only get scanned a constant number of times.

Following the spirit of DFS, we wish to maintain that the search structure is not self-intersecting, i.e., each vertex is visited at most once with an *inner* tag, and once with an *outer* tag. Whenever we discover an edge that leads to a self-intersection (e.g. (v_4, v_1) in Figure 3.6), we augment along the alternating cycle introduced by this edge (e.g. $(v_1, v_2, v_3, v_4, v_1)$) and thereby remove every edge on the cycle from eligible subgraph. We backtrack to v_1 and the search continues (to the edge (v_1, v_5)). This action has the same effect as allowing augmentation along the non-simple augmenting walk $((v_0, v_1, v_2, v_3, v_4, v_1, v_5, v_6))$, but conceptually avoids a self-intersecting search structure and thus makes the analysis much simpler.

Overview of the algorithm. In this algorithm, we follow a standard recursive framework for computing a maximal set of edge disjoint paths, see [132, §9]. The algorithm proceeds in phases. In phase i, $i \geq 1$, we choose a vertex r that is still unsaturated after augmentations in phase $1, 2, \ldots, i-1$, and call a procedure `SEARCH-ONE` from this vertex. `SEARCH-ONE` searches for an augmenting walk from r, and terminates by either discovering an augmenting walk P_i, or correctly concluding that no

augmenting walk can be found starting from r, in which case we let P_i be empty. Along the way it may find a set of alternating cycles $\mathcal{C}_i$. It then augments along the augmenting walk as well as the set of alternating cycles it encounters during the search. The phase ends by discarding the set of edges encountered by the search procedure.

Formally, the input to `SEARCH-ONE` is a subgraph $G_i = (V_i, E_i)$ of G, an f_i-matching M_i where $f_i(v) \leq f(v)$ for all $v \in V_i$ and $M_i \subseteq M$, and an unsaturated vertex $r \in V_i$ with respect to f_i and M. `SEARCH-ONE` finds a augmenting walk P_i (possibly empty), a set of alternating cycles $\mathcal{C}_i$ and a set of edges $H_i \subseteq E_i$ that satisfy the following property, in $O(|H_i|)$ time.

Property 3.36. *Any augmenting walk that intersects H_i must also intersect P_i or a cycle in $\mathcal{C}_i$.*

After `SEARCH-ONE` terminates, we terminate the phase by removing H_i from G_i. If P_i is empty, we also remove the vertex r from G_i. Let the resulting graph be G_{i+1}. Define the f_{i+1}-matching M_{i+1} by $M_{i+1} = M_i \setminus H_i$ and

$$f_{i+1}(v) = \begin{cases} f_i(v) - |M_i \cap H_i \cap \delta(v)| - 2|M_i \cap H_i \cap \delta_0(M_i)| & \text{If } v \text{ is not a terminal vertex of } P_i. \\ f_i(v) - |M_i \cap H_i \cap \delta(v)| - 2|M_i \cap H_i \cap \delta_0(M_i)| - 1 & \text{If } P_i \text{ is a nonempty non-closed augmenting walk and } v \text{ is one of the two terminal vertices of } P_i. \\ f_i(v) - |M_i \cap H_i \cap \delta(v)| - 2|M_i \cap H_i \cap \delta_0(M_i)| - 2 & \text{If } P_i \text{ is a nonempty closed augmenting walk that starts and ends with } v. \end{cases}$$

Conceptually, this change restricts the f_i-matching M_i to the subgraph G_{i+1} while maintaining the property that each vertex still has the same deficiency, except for the terminal vertices of P_i, whose deficiencies are decremented by 1 (or 2 for closed walks) after augmenting along P_i. Finally, the algorithm adds the path P_i and cycles $\mathcal{C}_i$ to

Ψ and $\mathcal{C}$, respectively, and terminates phase i.

A detailed illustration of `SEARCH-ONE`. The call to `SEARCH-ONE` in phase i maintains a laminar set of blossoms Ω_i over vertices in G_i. Here Ω_i only contains the blossoms newly discovered in the search procedure and does not include the already contracted blossoms inherited from the input (vertices in V_b). In this section, we use the word *blossom* solely for the newly discovered blossoms in Ω_i and *blossom vertices* for blossoms inherited from the input, i.e., vertices in V_b. *Singletons* still refer to vertices in V_s. We use $B(v)$ to denote the inclusion-wise maximal blossom in Ω_i that contains v and $\beta(v)$ to denote the base of $B(v)$. If v is not contained in any nontrivial blossom in Ω_i, we define $B(v) = \{v\}$ and $\beta(v) = v$. We denote the search structure on G_i with S_i, and use T_i to denote the search structure obtained from S_i by contracting all blossoms in Ω_i. Similar to [75, §8], the search structure S_i is a subgraph of G_i but not necessarily a tree, while we maintain that T_i must be a tree. Each blossom also might have a base edge $\eta(B)$. Base edge of a blossom, if exists, always connects it to its parent in T_i.

Blossoms are maintained using a data structure that supports the following operation: given a blossom B, a vertex v in blossom B, and a bit $s \in \{0, 1\}$, the data structure returns the alternating walk $P_s(v)$ from v to $\beta(B)$ whose existence is guaranteed in Lemma 3.2, in time linear in the length of the walk. This can be done using simple bookkeeping as in Gabow's implementation for Edmonds algorithm [68] and we leave the details to the readers.

`SEARCH-ONE` explores the graph in a depth first fashion: The search begins at an unsaturated singleton or an unsaturated blossom vertex r in G_i. Similar to DFS, when the locus of the search is at u we have an alternating walk $P(u)$ from r to u in G_i. We call u the *active vertex* and $P(u)$ the *active walk*. For efficiency purposes, we do not maintain the active walk explicitly. Instead, we maintain a contracted active walk $\widehat{P}(u)$. The contracted active walk $\widehat{P}(u)$ is of the form $\langle B_0, e_0, B_1, e_1, \dots, e_{k-1}, B_k \rangle$, where $r \in B_1$, $u \in B_k$ and each B_j are either singletons or blossoms (not necessarily maximal) in Ω_i. Each edge e_j connects B_{j-1} to B_j and edge e_j and e_{j+1} *alternates* at B_j for all $0 \leq j < k - 1$.

Moreover, we can without lost of generality assume the last element B_k on the contracted walk must be a trivial blossom. This is because in our algorithm when a new blossom is created, all edges with exactly one endpoint inside the blossom will be made eligible and explored. Therefore, any nontrivial blossom must have all its incident edges explored and exhausted and we can backtrack along the active walk

until we get to the trivial blossom on the walk.

The active walk $P(u)$ can be reconstructed from $\widehat{P}(u)$ in time $O(|P(u)|)$ using the blossom data structure mentioned above. Moreover, all blossoms in Ω_i are *outer* from the perspective of definition 3.22, i.e. every blossom's base edge, if it exists, always connects it to its parent. Moreover, given any vertex u, it also support finding $\beta(u)$ in constant time[73].

To maintain the property that the active walk is alternating, the algorithm also maintains tags for each vertex in S_i, which is either *inner* or *outer*, or both. When the search reaches the vertex u, u is labelled as outer if the active walk to u terminates with an matched edge in the case when u is a singleton, or the base edge when $u \in V_b$ is a blossom vertex. A vertex in a blossom can be simultaneously inner and outer.

Initially, the contracted active walk consists of a single vertex r, and r is labelled outer. At each iteration, the algorithm *explores* a new edge (u, v) incident to the active vertex u that is *eligible for* u with respect to its current tag. On exploring the edge (u, v), the algorithm does one of the following six steps depending on the location of v with respect to the search structure, and the tag v is carrying:

1. *Augmentation*: When v is an unsaturated singleton and (u, v) is unmatched, or v is an unsaturated blossom vertex, or when $v = r$ is the root of the search tree and the deficiency of v is at least 2, $P(u) \circ (u, v)$ forms an augmenting walk. We extend the active path with (u, v), terminate the search and set the active walk $P(v) = P(u) \circ (u, v)$ and $P_i = P(v)$. In this step, the edge (u, v) is considered *explored from* u.
2. *DFS Extension*: If v is not in the search structure S_i, and the condition for Augmentation is not satisfied, i.e., v is saturated or v is an unsaturated singleton and (u, v) is matched, add (u, v) to the search structure S_i and make v a child of $B(u)$ in T_i. Set the active vertex to v and extend the contracted active walk $\widehat{P}(v) = P(u) \circ (u, v)$ accordingly. Tag v inner if $v \in V_s$ and $(u, v) \notin M$, or if $v \in V_b$ and $\{(u, v)\} \neq \eta(v)$, and outer otherwise. In this step, edge (u, v) is considered *explored from* u.
3. *Blossom Formation*: If we fail to enter the previous two cases, v must be inside the search structure S_i. If $B(u) \neq B(v)$ and $B(v)$ is a descendant of $B(u)$ in T_i, and edge (u, v) is also eligible for v, then we can form a new blossom. The new blossom B consists of subblossoms on the T_i alternating path $\tilde{P}$ from $B(u)$ to $B(v)$. When we form this blossom, every *singleton* and *blossom vertex* in the T_i path from $B(u)$ to $B(v)$ becomes simultaneously inner and outer. It is then

necessary for the DFS procedure to visit these vertices again and explore a new set of eligible edges.

We add the blossom B to Ω_i and set $B(w) = B$ and $\beta(w) = \beta(u)$ for each vertex w that is inside a blossom in $\tilde{P}$ and label all these vertices simultaneously inner and outer. We also define the base edge of B to be the base edge of $B(u)$ if $B(u)$ is a nontrivial blossom, or the last edge in the active walk $P(u)$ otherwise. Now let $\langle B(u) = B_1, e_1, B_2, e_2, \ldots, e_{k-1}, B_k = B(v)\rangle$ be the tree path from $B(u)$ to $B(v)$. We extend the contracted active walk from $B(u)$ around the blossom and back to $B(u)$ as follows: First go from $B(u)$ to $B(v)$ via edge (u, v), then follow the tree path back to $B(u)$. The new contracted active walk $\widehat{P}(u)$ is $\widehat{P}'(u) \circ (u, v) \circ \langle B_k, e_{k-1}, B_{k-1}, \ldots, e_1, B_1\rangle$, where $\widehat{P}'(u)$ is the old contracted active walk from $B(r)$ to $B(u)$. Notice that this step can be implemented in $O(k)$ time where k, defined above, is the number of immediate subblossoms of B.

We then start exploring edges incident to vertices in B that become eligible for its endpoint in B during this step. As mentioned before, vertices that are already in some nontrivial blossom before contracting B are already both inner and outer and do not get any new eligible edges. Therefore, it suffices to only examine those nontrivial blossoms.

In this step, edges on the walk $\tilde{P}$ are regarded as explored *from endpoint closer to* v as we extend the contracted active walk back to $B(u)$, then exhausted after the search backtracks over the edge. Notice that these edges are already exhausted from the endpoint closer to u.

4. *DFS Retraction*: If every edge (u, v) eligible for u has already been explored, retract from u to its predecessor on the (contracted) active walk. If $u = r$ is the only vertex in the search path, terminate the search with $P_i = \emptyset$. Otherwise, let w ($B(w)$) be the parent of u in the (contracted) active walk. The edge (w, u) is now considered *exhausted from* w. This means that every edge eligible for u is recursively exhausted and no augmenting walk can be found by following the active walk via the edge (w, u). (It may still be possible to find an augmenting walk via edge (w, u) when the search visits u again in a blossom formation step and explore (u, w) from u).

5. *Cycle Cancellation*: If $B(v)$ is an ancestor of $B(u)$ and (u, v) is not eligible for v, we know that v must not be in any nontrivial blossom or (u, v) will also be eligible for v. Therefore, the tree path from $B(v) = \{v\}$ to $B(u)$, along with the

edge (u, v), forms an alternating cycle C. We add C to $\mathcal{C}_i$. Retract the active walk back to v. After this step, all edges $e \in C$ will be categorized as explored from *both endpoints*.

6. *Null Exploration*: This step includes all scenarios where we explore the edge (u, v) to no effect. This includes: when $B(v)$ is a descendant of $B(u)$ and (u, v) is not eligible for v; when $B(v)$ is an ancestor of $B(u)$ and (u, v) is eligible for v; or when (u, v) is a cross edge in T_i. In these cases, we ignore the edge (u, v) while still categorizing it as *exhausted from* u.

As stated above, each edge (u, v) along with an endpoint of it, say u, has one of three statuses at any point in the algorithm:

1. *Explored from* u: This means the search has visited the vertex u, extended the active walk from u to v using edge (u, v), in either Augmentation, DFS extension, Blossom Formation or Cycle Cancellation step.
2. *Exhausted from* u: This means the search has visited the vertex u, extended the search path to v via (u, v) and then backtracked to u in DFS Retraction or Null Exploration steps.
3. *Unexplored from* u: If (u, v) is not considered explored or exhausted from u, it is then unexplored from u. This means the search has either yet to visit u; or the search has visited u but never extended the active walk using the edge (u, v) because it is ineligible for u; or it is eligible but the search has yet to explore (u, v).

Finally, we specify that the edge set H_i is the set of edges that are explored or exhausted from at least one of its endpoints. Recall this is the set we remove from the graph G_i before termination of a phase. This completes our description of the `SEARCH-ONE` procedure.

Now we state the set of invariants satisfied by `SEARCH-ONE`.

Invariant 3.37.

1. Structural Invariant of S_i: *S_i consists of all vertices in G_i that are visited during the search. Every vertex in S_i is either inner, outer, or both. If v is an inner vertex in S_i, there exists an alternating walk from r to v that ends with an unmatched edge if $v \in V_s$ or a non-η edge if $v \in V_b$. If v is outer, the alternating walk terminates with a matched edge if $v \in V_s$ or an η edge if $v \in V_b$.*
2. Structural Invariant of T_i: *T_i is a contracted graph obtained from S_i by contracting all inclusionwise-maximal blossoms in Ω_i. T_i must be a tree.*

3. Depth-first property of S_i: *The union of the active walk and the set of alternating cycles $\mathcal{C}$ consists of precisely the edges that are explored but not exhausted from at least one endpoint. If (u, v) is an edge in H_i but not in the active walk or alternating cycles, then (u, v) must be exhausted from u.*

4. Maximality of S_i: *If (u, v) is marked exhausted from u while (v, w) is an edge eligible for v, then the algorithm must have exhausted (v, w) from v.*

Lemma 3.38. Augmentation, DFS Extension, Blossom Formation, DFS Retraction, Null Exploration *and* Cycle Cancellation *all preserve Invariant 3.37.*

Proof. The first invariant follows from how we grow the search structure S_i and active walk. When the active walk extends to a vertex v with the current tag outer, the active walk must be an alternating walk ending with a matched edge or an η edge. If the tag is inner, the active walk ends with a unmatched edge or an non-η edge. This ensures that there exists an alternating walk from the root to each vertex in S_i with a terminal edge corresponding to its tag.

The second invariant follows from the fact that when we form a blossom in the Blossom Formation step, the constituent (subblossoms) in Ω_i always come from a connected ancestor-descendant path in T_i. Contracting a connected component in the tree will not create any cycle and thus T_i remains a tree.

For the third invariant, observe that an edge becomes explored from an endpoint when it joins the active walk in a DFS Extension, Blossom Formation, or Augmentation step. It becomes exhausted when it leaves the active walk at DFS Retraction, Null Exploration, and Cycle Cancellation step. Moreover, in the Blossom Formation step, since we are visiting vertices in descendant-to-ancestor direction, all edges in the active walk must remain explored and edges outside active walk are exhausted. Therefore, any edge in H_i that is not in the active walk must be exhausted.

For the fourth invariant, first notice that (u, v) becomes exhausted via a DFS Retraction step or a Null Exploration step. In both cases the search must have retracted from v to some vertices and therefore has explored and exhausted every edge eligible for v, including (v, w). If w is an unsaturated singleton and (v, w) is unmatched, or w is an unsaturated blossom vertex, an Augmentation step would have occurred when the algorithm explores (v, w) and left the edge (v, w) explored and not exhausted. □

The next lemma states that if one edge is once explored/exhausted from both directions, then both endpoints must be in the same blossom in Ω'. This is analogous to Property (i) in [75, §8].

Lemma 3.39. *If (u, v) is an edge that is once explored from both u and v, then u and v must be in the same blossom in Ω'.*

Proof. Notice that in depth-first search, when the algorithm explores (u, v) from both directions, $B(u)$ and $B(v)$ must be ancestor/descendant of one another in T_i. Now without loss of generality, we assume (u, v) is first explored from u. If v is a descendant of u, since the search has already backtracked from v, the only way that v enters the search again is by a blossom formation step from ancestor of u to a descendant of v, making them in the same blossom. If v is an ancestor of u, when (u, v) is explored from v, i.e., when the search backtracks from u to v, u must still be a descendant of v because any blossom step in this process will not change the ancestor-descendant relation between u and v. Then we have a blossom step triggered by (u, v) and put them in the same blossom. □

Now we state the correctness of `SEARCH-ONE`:

Lemma 3.40. *When `SEARCH-ONE` terminates, if there is an augmenting path P' that intersects H_i at some edge e, then P' must intersect P_i or $\mathcal{C}_i$ at some edge.*

Proof. Assume for contradiction that P' is edge-disjoint with P_i and $\mathcal{C}_i$. Let P' intersect H_i at some edge (u_0, u_1). Since (u_0, u_1) is not in P_i or $\mathcal{C}_i$, (u_0, u_1) must be exhausted in one of its directions, say from u_0. This makes (u_0, u_1) eligible for u_0. Now let $(u_0, u_1, \ldots, u_k)$ be the subpath of P' from u_0 to the terminal vertex u_k of P' in the (u_0, u_1) direction. We use induction to show that for all $0 \leq i < k$, edges (u_i, u_{i+1}) must be eligible for u_i and exhausted from u_i:

The base case $i = 0$ holds by our assumption. Now suppose (u_i, u_{i+1}) is exhausted from u_i for some $i \geq 0$. Consider the edge (u_{i+1}, u_{i+2}). It is necessary that u_{i+1} be in the search structure S and is either inner or outer or both. By the alternation requirement in Definition 3.7 and the definition of eligibility, either (u_i, u_{i+1}) or (u_{i+1}, u_{i+2}) (or both) must be eligible for u_{i+1}. If (u_{i+1}, u_{i+2}) is eligible for u_{i+1}, then by Invariant 3.37, (u_{i+1}, u_{i+2}) must be exhausted from u_{i+1}.

If (u_i, u_{i+1}) is eligible for u_{i+1}, by Invariant 3.37 (3), edge (u_i, u_{i+1}) must be exhausted from both directions. By Lemma 3.39, they must be in the same blossom and must have both inner outer tags. This makes (u_{i+1}, u_{i+2}) also eligible for u_{i+1}, and thus must be exhausted from u_{i+1}.

This means the edge (u_{k-1}, u_k) must be eligible for u_{k-1} and exhausted from u_{k-1}. Notice that the vertex u_k and edge (u_{k-1}, u_k) must satisfy the terminal vertex and edge requirement in Definition 3.7. But in this case, an augmenting walk would have

been formed when the algorithm was exploring the edge (u_{k-1}, u_k) from u_{k-1}, which put the edge in P_i, which is a contradiction. □

This gives the following Lemma:

Lemma 3.41. *`SEARCH-ONE` finds in time $O(|H_i|)$ an edge set H_i, a set of alternating cycles $\mathcal{C}_i$ and an augmenting walk P_i such that any augmenting walk P' disjoint from $\mathcal{C}_i$ that intersects H_i must also intersect P_i*

Proof. The correctness of `SEARCH-ONE` is argued in Lemma 3.40. For running time, notice that each edge we examined is always classified as explored or exhausted from at least one of its endpoints. Thus, the total number of edge examinations is $O(|H_i|)$. The only non-trivial data structure needed is one for maintaining the set of blossoms, in particular $\beta(\cdot)$, which can be solved in $O(n + m\alpha(m, n))$ with the standard union-find algorithm [134] or in optimal $O(m+n)$ time with the incremental-tree union-find algorithm of Gabow and Tarjan [73]. For reconstructing the active walk, we can use the bookkeeping labelling in [75, §8], which enable reconstruction of the augmenting walk in time linear in terms of the length of the walk. □

Lemma 3.42. *We can find in linear time a set of augmenting walks Ψ and a set of alternating cycles $\mathcal{C}$ such that any augmenting walk P' must intersect Ψ or $\mathcal{C}$.*

Proof. This algorithm can be seen as a recursive algorithm that first calls `SEARCH-ONE` on an input graph $G_1 = G$, finding an edge set H_1, a set of alternating cycles C_1 and an augmenting walk P_1. It removes H_1 from G_1 and the corresponding part in the f-matching to obtain G_2. Then it recurses on G_2. Let $\mathcal{C}'$ and Ψ' be the output of the recursive call. We output $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}'$ and $\Psi = \Psi' \cup \{P_1\}$.

By induction, any augmenting walk P' in G_2 must intersect Ψ' or $\mathcal{C}'$. Now suppose the augmenting walk P' contains an edge in H_1. By Lemma 3.41, P' must intersect P_1 or $\mathcal{C}_1$. Therefore P' must intersect Ψ or $\mathcal{C}$. □

Finding a maximal set of nested blossoms We can also show that, when the graph G does not contain any augmenting walk, the union $\Omega' = \bigcup_i \Omega_i$ forms a maximal set of nested blossoms. The maximality is characterized by Lemma 3.39: a blossom step can be performed only if we discover an edge (u, v) that is eligible for both u and v while searching from a same root. By Lemma 3.39, we would have already contracted it into one blossom. This means no more reachable and contractable blossom exists after we contract Ω'.

We also need to argue that Ω' is a laminar family, in particular, blossoms from Ω_i and Ω_j for $j > i$ does not intersect. This is because when we form a blossom in Ω_i, all its constituents must have all their incident edges explored and thus removed from any future searches and thus any future blossoms.

3.6 Algorithms for Unweighted f-Matching and f-Edge Cover

In this section we will give an $O(\sqrt{f(V)}m)$ algorithm for both maximum cardinality f-matching and minimum cardinality f-edge cover. This is a direct consequence of the $O(Wm\epsilon^{-1})$ algorithm for the weighted problem. This algorithm matches the running time of [69] but does not rely on reduction to iterations of the Micali-Vazirani algorithm [110, 137, 138]. Moreover, it is much simpler to state and to analyze.

For illustration purposes we focus on maximum cardinality f-matching. The algorithm consists of two phases. The first phase, referred to as *batch augmentation*, finds an f-matching F that is close to optimal using an instance of the $O(Wm\epsilon^{-1})$ algorithm. After F is close to optimum, we discard all dual variables y and z, dissolve all the blossoms in Ω and use our linear time augmenting walk algorithm to increase the cardinality of F until F becomes optimum.

This is stated formally in the following theorem:

Theorem 3.43. *A maximum cardinality f-matching can be computed in $O(\sqrt{f(V)}m)$ time.*

Proof. We can view the maximum cardinality f-matching problem as a maximum weight problem with weight function $w(e) = 1$. Choose $\epsilon = 1/\sqrt{f(V)}$, by Theorem 3.26, we can compute a $(1 - \frac{1}{\sqrt{f(V)}})$-approximate maximum cardinality matching F in $O(\sqrt{f(V)}m)$ time. If F^* is the maximum cardinality f-matching, we have

$$|F| \geq \left(1 - \frac{1}{\sqrt{f(V)}}\right)|F^*| > |F^*| - \frac{1}{\sqrt{f(V)}} \cdot \frac{f(V)}{2} > |F^*| - \sqrt{f(V)}/2.$$

This means F is only $O(\sqrt{f(V)})$ augmentations away from optimal. Hence we can then discard the blossom structure Ω with duals y and z from the approximate f-matching and run the linear time augmenting walk algorithm of Lemma 3.42 in G

with respect to F until F is optimal. By the discussion above, $O(\sqrt{f(V)})$ iterations suffice. The total running time of the algorithm is $O(\sqrt{f(V)}m)$. □

Theorem 3.44. *A minimum cardinality f-edge cover can be computed in $O(\sqrt{f(V)}m)$ time.*

Proof. This is similar to Theorem 3.43. We first use the $O(Wm\epsilon^{-1})$ algorithm for f-edge cover in Corollary 3.27 to find an $(1+\sqrt{f(V)}^{-1})$-approximate minimum cardinality f-edge cover F by viewing the graph as a weighted graph with weight 1 everywhere. Choosing $\epsilon = 1/\sqrt{f(V)}$ will give us an f-edge cover F with $|F| \leq |F^*|+\sqrt{f(V)}^{-1}|F^*|$. Notice that we always have $|F^*| \leq f(V)$ because taking $f(v)$ arbitrary incident edges for each v and taking their union will always give a trivial f-edge cover with cardinality at most $f(V)$. Hence we have $|F| \leq |F^*| + \sqrt{f(V)}$, which means at most $O(\sqrt{f(V)})$ reductions are needed to make F optimal. Therefore we can run the augmenting path algorithm from Lemma 3.42 to find reducing paths (P is a reducing path w.r.t. F if and only if it is an augmenting path w.r.t. $E \setminus F$) until no reducing path can be found. There are $\sqrt{f(V)}$ iterations in this phase. The total running time of the algorithm is $O(\sqrt{f(V)}m)$. □

3.7 Conclusion and Open Problems

We give the first almost linear time approximation schemes for both MAXIMUM WEIGHT f-MATCHING and MINIMUM WEIGHT f-EDGE COVER PROBLEM. This result generalizes the algorithm by Duan and Pettie [55] for approximate weighted matching problems. We also establish approximation preserving reductions between the two problem inside the primal-dual framework for weighted matching, showing that any approximation algorithm that also produces an approximately optimal dual solution to one problem can be transformed to an approximation algorithm to the other problem.

We demonstrate that with the appropriate generalization of key concepts such as blossoms and augmenting paths, we can derive direct algorithm for generalized matching problems from its corresponding version for 1-matching, instead of relying on inefficient reductions. Thus, it is natural to ask whether other algorithm for 1-matching, such as [57] can be generalized to f-matching.

Another key open problem is whether we can obtain linear dependency in graph size for our algorithm, i.e. eliminating the factor $\alpha(m, n)$ in the time complexity. The

key challenge here is to give a union find data structure for maintaining blossoms on a dynamic tree as specified in Section 3.5.

CHAPTER 4

Metric Matching

4.1 Introduction

The MINIMUM WEIGHT PERFECT MATCHING PROBLEM (MWPM) is another important problem in matching theory. It asks for minimum weight solution among the set of all *perfect* matching. The MWPM and MWM problems are known to be reducible to each other. Given an algorithm for MWM in with running time $f(n, m, W)$, it can be used to solve MWPM in time $f(n, m, nW)$ time. On the other hand, given an MWPM algorithm in time $g(n, m, W)$, we can obtain an algorithm for MWM in time $g(O(n), O(m), W)$. As a result, most algorithmic results for weighted matching [60, 76, 70, 75, 47, 56] apply to both problems.

However, since the reduction above from MWPM to MWM is not approximation-preserving, similar conclusion does not hold for approximate-MWPM and approximate-MWM. In particular, approximate-MWM in almost linear time has a long history [122, 53, 55] starting from a greedy algorithm which give a 1/2-approximation. However, there is no known approximate-MWPM algorithm in almost linear time for general graph. This is because an approximate-MWPM needs to be a *perfect* matching, so we cannot hope for a linear time approximation algorithm without improving the Micali-Vazirani algorithm for maximum cardinality matching, which resists any improvement for 40 years. Moreover, even for graphs where a perfect matching is easy to compute, such as complete graph with arbitrary weight function, an finding a $o(W/n)$-approximate solution is as hard as computing a perfect matching in a general graph.

However, many of the high profile applications of MWPM problem are not for general weighted graphs, but the distance metric induced by an underlying graph. For example, in solving the *Chinese Postman Problem*[132], one needs to find a min cost perfect matching between the odd degree vertices to make a graph Eulerian. In

Christofides' well known 3/2-approximation algorithm for metric TSP problem, we compute a min weight perfect matching among a subset of vertices where the cost function is induced by distances in the graph. The canonical way to solve the problem is to run any APSP algorithm to obtain the distance matrix, and then run any exact MWPM algorithm, say [56]. This makes computing perfect matchings the bottleneck for a near linear time implementation for Christofides algorithm [33]. Alternatively, one can run the 2-approximation algorithm by Goemans and Williamson[78] and its almost linear time implementation by Cole et al. [40] or Gabow and Pettie [67] for dense graph. We are not aware of any alternate solution to this problem for obtaining a near linear time implementation for Christofides algorithm with ϵ-sacrifice in approximation ratio. Alternatively, one can achieve a $(3/2+\epsilon)$ approximate by first applying an LP based cut sparsifier before computing the metric perfect matching [34].

Specifically, for metrics induced by geometric spaces, e.g., points in Euclidean plane, Varadarajan and Agarwal [136] gave an almost linear time approximation scheme that returns a $(1+\epsilon)$-approximate solution in $O(n\epsilon^{-3}\log^6 n)$ time. The algorithm is based on the plane decomposition scheme by Arora [16].

Problem Statement and Terminologies. In this paper, we consider the METRIC MIN COST PERFECT MATCHING PROBLEM where the metric is induced by an *unweighted and undirected* graph G and a subset of terminals T. The problem can be defined as follows: Let $G = (V, E)$ be an unweighted and undirected graph, and $T \subseteq V$ be an even set of terminals. We use n and m to denote the number of vertices and edges respectively. Fix the set of terminals T, let $\mathcal{G}$ be the *metric completion* induced by the graph G and the terminal set T, i.e., $\mathcal{G} = (T, \binom{T}{2}, d_G)$ is a weighted complete graph on vertex set T and weight function d_G which is the distance function for G restricted to $T \times T$.

Metric Matching and T-join The GRAPH METRIC MIN WEIGHT PERFECT MATCHING PROBLEM (abbr. Graphic-MWPM) ask the following question: Given an unweighted and undirected graph $G = (V, E)$ and an even set of terminal $T \subseteq V$, find the minimum weight perfect matching in the metric graph $\mathcal{G}$.

This problem is equivalent to the *T-join* problem. Given a graph $G = (V, E)$ and a subset T of vertices, a *T-join* is a subset $F \subset E$ of edges such that in the subgraph (V, F), T is precisely the subset of vertices with odd degree. The MINIMUM CARDINALITY T-JOIN PROBLEM ask for the following: Given an unweighted graph $G = (V, E)$ and an even subset $T \subseteq V$ of vertices, find a T-join in G of minimum

size.

It can be shown that the two problem are equivalent [33] for even weighted graph. More precisely, a minimum weight T-join consists of $|T|/2$ edge disjoint shortest paths between disjoint pairs of terminals in T which forms a perfect matching. And for any minimum weight perfect matching in $\mathcal{G}$ consists of $|T|/2$ edge disjoint shortest paths between pairs of terminals in T which also forms a T-join.

Notice that if n is even and we set $T = V$, the problem is equivalent to the MWPM problem on unweighted graph.

Growth Expansion property is an important concept in algorithmic study of metric space [5, 102, 87, 94, 25, 20]. It captures the geometry of the underlying metric space via characterizing the relation between the volume and diameter. Many problems such as nearest neighbor [94] or travelling salesman [20] are hard for general metrics since general metrics simply do not provide enough geometric structure for exploitation. Yet instances that arise from practice, such as database queries or geometric optimization are often far from general, and expansion, growth and dimensionality are used to characterize the geometric properties of these instances.

Here, we define the growth as the number of vertices you see from a vertex's neighborhood as a function of the neighborhood radius. More formally, in a metric space (V, d), the *ball* $B(u, r)$ with center u and radius r is the set $\{v \in V : d(u, v) \leq r\}$, denoted by $B(u, r)$. We define the growth of the graph G as the *minimum* size of $B(u, r)$ as a function of r.

Definition 4.1. *Let $f : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ be a nondecreasing nonnegative function. We say G has* growth f *if for all $v \in V$ and $r \geq 1$, $|B(v, r)| \geq \min\{f(r), n\}$.*

For example, an unweighted line metric has linear growth r, while a d-dimensional grid has growth r^d.

Treewidth *Treewidth* is a fundamental graph parameter that measures how well a graph resembles a tree. In particular, it measures how well a graph can be decomposed in a treelike manner. Similar to growth, many graphs from practical instances are known to have bounded treewidth [109], and many problems are known to be efficiently solvable on graphs with small treewidth, such as reachability and shortest paths [12, 28, 121], multi-commodity flow [82] and various matching and matrix problems [65]. Many classical NP-hard problems such as vertex cover and dominating set also admit polynomial algorithms for graphs with bounded treewidth; see [46]. Treewidth can be defined from a *tree decomposition* as follows:

Definition 4.2. *A* tree decomposition *of a graph G is a pair $(\mathcal{T}, \{B_x\}_{x \in V(T)})$, where $\mathcal{T}$ is a tree and each node x in $\mathcal{T}$ is associated with a set of vertices $B_x \subseteq V(G)$, called the* bag *of x, that satisfies the following conditions:*

1. *For each edge $(u, v) \in E(G)$, there exists $x \in V(\mathcal{T})$ such that $\{u, v\} \subset B_x$.*
2. *For each $u \in V(G)$, let $\mathcal{T}[u]$ be subtree of $\mathcal{T}$ induced by the set of bags containing u. Then $\mathcal{T}[u]$ is nonempty and connected.*

Given a tree decomposition $(\mathcal{T}, \{B_x\})$, its width can be defined as:

Definition 4.3. *The* width *of $(\mathcal{T}, \{B_x\})$ is $\max_{v \in V(T)} |B_x| - 1$.*

The tree width of a graph G is the minimum width of any tree decomposition of G.

It is NP-hard to compute the treewidth of a graph exact [15]. However, there are plenty of FPT and approximation algorithms that compute an optimal or near optimal tree decomposition. For instance, the approximation algorithm by Fomin et al. [65] suits our study:

Theorem 4.4 ([65] Theorem 1.1)**.** *There is an algorithm that given a graph G of n vertices and a positive integer k, in time $O(k^6 n \log n)$ either finds a tree decomposition of width $O(k^2)$, or correctly concludes the treewidth of G is at least k.*

Matching has been studied in graphs with bounded treewidth. Courcelle's celebrated theorem [44] states that any (optimization variant of) decision problems expressible with *Monadic Second Order Logic* admits a $\tilde{O}(f(k)n)$ algorithm for some function f if we are given a tree decomposition of width k. For the maximum matching problem, it is not hard to obtain an algorithm running in time $\tilde{O}(3^k n)$. Fomin et al. [65] gives an algorithm for MWM and MWPM with time complexity $O(k^3 n \log n)$. However, we are not aware of FPT algorithm for Graphic-MWPM for graphs with bounded treewidth.

Our Contribution: In this paper, we show that the graphic-MWPM admits a linear time approximation scheme if the graph has slightly superlinear growth:

Theorem 4.5. *Let $G = (V, E)$ be an unweighted and undirected graph that has n vertices and m edges, and growth $f(d) \geq d \log^2 d$. Let $T \subseteq V$ be a set of terminals and $d : T \times T \mapsto \mathbb{Z}_{\geq 0}$ be the distance metric on G restricted to T. Then an ϵn additive approximate solution to the graphic-*MWPM *problem can be computed in time $O(m \log n 2^{O(1/\epsilon)})$ time. For a graph with polynomial growth, i.e., $f(d) \geq d^{1+\tau}$, an ϵn additive approximate solution can be obtained in time $O(m \log n (1/\epsilon)^{1/\tau})$.*

We also show that if we are also given a tree decomposition of width k, we can solve the graphic-MWPM in $O(mk \log^2 n)$ time, improving the result of Fomin et al [65] on unweighted graph.

Theorem 4.6. *Given an unweighted and undirected graph $G = (V, E)$ along with a tree decomposition of width k, and a set of terminals $T \subseteq V$, we can solve the graphic-*MWPM *problem in time $O(mkp \log n)$, where p is the update amortized time for a priority queue that supports insert and delete-min.*

Organization In Section 4.2, we are going to review the LP framework of Edmonds for matching and perfect matching. Section 4.3 will first review the main algorithmic component of our approximate min cost perfect matching algorithm, Edmonds' Search. It then presents our scaling algorithm for approximate min cost perfect matching algorithm on graphs with moderate growth. Section 4.4 will solve the data structure problem left in Section 3, namely the implementation of Edmonds's Search implicitly on a graph metric. In Section 4.5, we present our algorithm for computing exact min cost perfect matching for metrics generated by bounded treewidth graphs.

4.2 Primal, Duals and Complementary Slackness

In this section we review Edmonds' primal-dual framework for weighted matching. The min weight perfect matching problem can be formulated as the following LP:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{e \in \delta(u)} x_e = 1, \quad \forall u \in V \\ & \sum_{e \in \delta(B)} x_e \geq 1, \quad \forall B \subset V, |B| \text{ odd} \\ & 0 \leq x_e \leq 1, \quad \forall e \in E \end{aligned} \tag{4.1}$$

For the corresponding dual program, we have a dual variable y associated with each vertex and a dual variable z with each odd set, formulated as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} y_v + \sum_{B \subseteq V, B \text{ odd}} z_B \\ \text{subject to} \quad & y_u + y_v + \sum_{B : (u,v) \in \delta(B)} z_B \leq w(u,v), \quad \forall (u,v) \in E \\ & z_B \geq 0, \qquad \forall B \subseteq V, |B| \text{ odd} \end{aligned}$$

In the algorithm we maintain a specific set of *blossoms* $\Omega \subset 2^T$. This is the set of odd sets that could have positive z-values during the algorithm. We define an aggregate dual on a single edge (u, v) to be:

$$yz(u, v) = y(u) + y(v) + \sum_{B \in \Omega : |B \cap \{u,v\}| = 1} z_B$$

And the aggregate dual for a vertex u is:

$$\lambda(u) = y(u) + \sum_{B \in \Omega : u \in B} z(B)$$

Observe that in general, we have $yz(u, v) \leq \lambda(u) + \lambda(v)$. The equality holds if there is no blossom in Ω containing both u and v.

4.2.1 Blossoms

During the algorithm certain odd sets of vertices are designated as *blossoms*. Formally, a blossom can be inductively defined as follows: A blossom is a set of vertices B associated with an edge set E_B. A single vertex v forms a trivial blossom,[1] or *singleton*. Its edge set is the empty set. Suppose for some odd $l \geq 3$, $B_0, B_1, \ldots, B_{l-1}$ are disjoint blossoms. Then we call their union $B = \bigcup_{i=0}^{l-1} B_i$ a blossom if there exists edges and an alternating cycle $C = \langle e_0, e_1, \ldots, e_{l-1} \rangle$ such that $e_i \in A_i \times A_{i+1} \pmod{l}$ and e_i is matched if and only i is odd. The associated edge set is $E_B = (\bigcup_{i=0}^{l-1} E_{B_i}) \cup \{e_0, e_2, \ldots, e_{l-1}\}$. Here we call B_i's the *subblossoms* of B. If Ω is a laminar family of blossoms, we can represent the inclusion relation of blossoms with a collection of *blossom trees*, or *blossom forest*. The leaves are singletons and all nontrival blossoms have their subblossoms as their children. We refer to the root blossoms as the *outermost* blossom. For a terminal u, we use $B(u)$ to denote the outermost blossom that contains u.

Each blossom also has a designated vertex known as the *base*. The base of a singleton is itself. For blossom $B = \bigcup_{i=0}^{l-1} B_i$ defined inductively as above its base is inductively the base of B_0. We call a blossom *matched* if its base is matched and *free* if its base is free.

If Ω is a laminar family of blossoms, the *contracted graph* G/Ω is obtained by contracting all outermost blossoms into a single vertex. To *dissolve* an outermost blossom means to remove the blossom from Ω and replace it with all its immediate

[1] For simplicity from this point we will only refer to nontrivial blossom as blossom.

subblossom in the contracted graph. A crucial property for blossoms is that any augmenting path in the contracted graph can be extended to an augmenting path in original graph. We refer readers to [59, 75] for details.

4.2.2 Complementary Slackness

Similar to matching algorithms using Edmonds' primal-dual frameworks [55, 75, 136], we characterize the optimality with the following approximate complementary slackness property.

Property 4.7 (Approximate Complementary Slackness)**.** *Let $\Delta \geq 0$ be an error parameter and $d(u, v)$ the edge weight function:*

1. *Domination. For all $u, v \in T$, $yz(u, v) \leq d(u, v)$.*
2. *Approximate Tightness. For all $(u, v) \in M \cup \bigcup_{B \in \Omega} E_B$, $yz(u, v) \geq d(u, v) - \Delta$.*
3. *Nonnegative Dual. The z-values of all blossoms in Ω are nonnegative.*
4. *Structural Blossoms. Ω forms a laminar family of blossoms and for each blossom $B \in \Omega$, the matching M is maximal inside B, i.e. $|M \cap E(B)| = (|B| - 1)/2$.*

The main task of our algorithm is to maintain Property 4.7 without direct access to $\mathcal{G}[T]$. To present all the duals explicitly on G, we introduce notion of *dual balls*. For any terminal u, we define the dual ball of u being the set $\mathcal{B}_u = \{v \mid d(u, v) \leq \lambda(u)\}$. For any outermost blossom $B \in \Omega$, the dual ball of B is the union of the dual balls of all its vertices.

A single dual ball of a blossom B is represented in our algorithm by a breath first search forest $\mathcal{F}_B$. The roots of $\mathcal{F}_B$ are the terminals in B and we denote the tree rooted at u as $\mathcal{T}_u$. In principle $\mathcal{T}_u$ extends to all vertices in $\mathcal{B}_u$ but this will lead to intersections of distinct trees and thus is too expensive to maintain. Hence when growing the tree we retract from all but one tree whenever multiple tree intersect at one vertex. As we will see in next section this still allows us to correctly maintain $\mathcal{B}_B$, whenever the dual variables changes.

Notice that if d is the exact distance function and y, z is a set of duals that satisfy Property 4.7, dual balls of distinct outermost blossoms will only intersect at their boundaries, i.e., their intersection vertices must be leaves of the corresponding trees.

4.3 Scaling for Approximate Min weight Perfect Matching

In this section we first review Edmonds' Search, the main algorithm component of our scaling algorithm. We note that similar search procedures are studied in the context of maximum weight matching [75, 55, 56], for min weight perfect matching with a different LP formulation [75] or different complementary slackness criterion [136]. However we are not aware of any study of Edmonds' Search under precisely our setting. The structure of our proof is similar to that of [55].

4.3.1 Edmonds' Search in a nutshell

Conceptually, Edmonds' Search operates on a subgraph of $\mathcal{G}$ defined by a subset of *eligible* edges. In the metric graph $\mathcal{G}$, we call an edge (u, v) *eligible* if it satisfies one of the following:

1. $(u, v) \in E_B$ for any $B \in \Omega$.
2. $(u, v) \notin M$ and $yz(u, v) = d(u, v)$.
3. $(u, v) \in M$ and $yz(u, v) = d(u, v) - \Delta$.

The *eligible subgraph* $\mathcal{G}_{\text{elig}}$ of the metric graph contains only eligible edges. We use $\overline{\mathcal{G}_{\text{elig}}}$ to denote the *contracted* eligible graph $\mathcal{G}_{\text{elig}}/\Omega$ obtained by contracting all outermost blossoms in Ω. To contrast them we will refer to vertices in $\mathcal{G}_{\text{elig}}$ as vertices and vertices in the contacted graph as *nodes*. Notice that the asymmetric eligibility criterion for matched edges and unmatched edges are motivated by the fact that every edge in an eligible augmenting path becomes ineligible if we augment the matching along this path. This will be formally stated later in this section.

Let $\Delta \geq 0$ be some granularity parameter and suppose d is a distance metric such that all distances are multiples of Δ. One iteration of Edmonds' Search starts with some matching M and duals y, z, Ω that satisfy Property 4.7 as well as the following property:

Property 4.8 (Granularity)**.** *All y-values and z-values are multiple of* $\Delta/2$ *Furthermore, the* λ*-values of all free vertices are the same.*

The algorithm operates in iterations. Each iteration consists of four steps: *Augmentation*, *Blossom Formation*, *Dual Adjustment*, and *Blossom Dissolution*.

1. *Augmentation.* Find a maximal set Ψ of vertex disjoint eligible augmenting paths in $\mathcal{G}_{\text{elig}}$. Update the matching $M \leftarrow M \oplus \bigcup_{P \in \Psi} P$. Update $\mathcal{G}_{\text{elig}}$.

2. *Blossom Formation.* Let $\overline{T_{out}}$ be a the set of nodes reachable from a free node via an even length alternating path. Find a *maximal* set of nested blossoms Ω' on $\overline{T_{out}}$. The maximality here means after contracting blossoms in Ω', the contracted graph does not contain any blossom reachable via an eligible alternating path. Update $\Omega \leftarrow \Omega \cup \Omega'$. Set $z(B) \leftarrow 0$ for any blossom in Ω'. $\mathcal{G}_{\text{elig}}$ does not change in this step.
3. *Dual Adjustment.* Let $\overline{T_{in}}$ be the set of nodes not in $\overline{T_{out}}$ still reachable from a free nodes via an eligible (odd length) alternating path. Let T_{out} and T_{in} be the set of terminals in T that is contained in some nodes in $\overline{T_{out}}$ and $\overline{T_{in}}$ respectively. Change the dual variables as follows:

$$\begin{aligned}
y(u) &\leftarrow y(u) + \Delta/2 && \text{if } u \text{ is in } T_{out} \text{ but not in any blossom} \\
z(B) &\leftarrow z(B) + \Delta/2 && \text{if } B \text{ is outermost blossom in } T_{out} \\
y(u) &\leftarrow y(u) - \Delta/2 && \text{if } u \text{ is in } T_{in} \text{ but not in any blossom} \\
z(B) &\leftarrow z(B) - \Delta/2 && \text{if } B \text{ is outermost blossom in } T_{in}
\end{aligned}$$

   Update $\mathcal{G}_{\text{elig}}$.
4. *Blossom Dissolution.* After dual adjustment some outermost blossoms have 0 z-values. Dissolve those blossom as long as they exists. Update Ω and $\mathcal{G}_{\text{elig}}$.

We will also refer to vertices/nodes in T_{out} as *outer* vertices and vertices/nodes in T_{in} as inner vertices/nodes. The correctness of Edmonds' Search is stated in the following Lemma:

Lemma 4.9. *Suppose d is a weight function whose values are all multiple of Δ. If M, y, z, Ω are primal and duals that satisfy Property 4.7 and Property 4.8, they continues to satisfy Property 4.7 and Property 4.8 after one iteration of Edmonds' Search.*

One can prove the following standard fact that after augmenting along a maximal set of augmenting path, the eligible subgraph $\mathcal{G}_{\text{elig}}$ contains no augmenting path. Similarly after the blossom formation step there is no contractable blossom in $\mathcal{G}_{\text{elig}}$ reachable from a free vertex.

Lemma 4.10 ([55, 75])**.** *After augmentation and blossom formation, $\mathcal{G}_{\text{elig}}$ contains no augmenting path, nor any blossom reachable from a free vertex via an alternating path.*

The correctness of dual adjustment step also relies on the following parity lemma, which states that the λ-values of all reachable vertices in the eligible subgraph have the same parity:

Lemma 4.11. *Let $\overline{T_r}$ be the set of nodes that is reachable from a free nodes via an eligible alternating path, and T_r be its image in $\mathcal{G}_{\text{elig}}$. Then the λ-values of all vertices in T_r have the same parity as a multiple of $\Delta/2$.*

Proof. By definition of eligibility, the *slack* on each eligible edge, $d(u,v) - yz(u,v)$, must be an integer multiple of Δ. Suppose λ-value of all vertices in the same outermost blossom has the same parity as multiples of $\Delta/2$, then adjacent blossoms in $\overline{\mathcal{G}_{\text{elig}}}$ must have all of their terminals having the same λ-values parities. Furthermore, since each blossom is initially formed using only eligible edges, we can show by induction on nested blossoms that all vertices in the same outermost blossom must have the same parity in their λ-values. This finishes the claim. □

Lemma 4.12. *Suppose the values of the distance function d are all multiples of Δ. If M, y, z, Ω satisfy Property 4.7 and Property 4.8 before an iteration of Edmonds' Search, then they continue to satisfy Property 4.7 and Property 4.8 after Edmonds' Search.*

Proof. For Property 4.8, the only place that it can be violated is at dual adjustment. Notice that the net change in λ value for a vertex u is $+\Delta/2$ if $u \in T_{out}$, $-\Delta/2$ if $u \in T_{in}$ and 0 if u is unreachable. Since all the free vertices are in T_{out}, Property 4.8 is preserved.

For Property 4.7, we discuss its invariance by steps. In augmentation, Property 4.7 is preserved because we only augment along eligible augmenting paths, which satisfy $yz(u,v) \leq d(u,v)$ for matched edges and $yz(u,v) \geq d(u,v) - \Delta$ for unmatched edges. After they exchange status they satisfy both domination and approximate tightness, hence preserving Property 4.7.

Similarly, since the slack $yz(u,v) - d(u,v)$ of eligible edges satisfies the criterion for both domination and approximate tightness, blossom formation step also preserves Property 4.7. The nonnegative dual property is not endangered at both steps.

For dual adjustment, let (u,v) be an edge. If both of u and v are not reachable (outside $T_{in} \cup T_{out}$), then $yz(u,v)$ remains unchanged. Similarly if both u and v are in the same outermost blossom, $yz(u,v)$ remains unchanged.

Hence we now assume u, v are in distinct outermost blossoms and we have $yz(u,v) = \lambda(u) + \lambda(v)$. Suppose one of u and v, say u is reachable and v is not. Suppose

$u \in T_{out}$, hence (u, v) is unmatched otherwise v must be the unique predecessor vertex in T_{in}, making v reachable. Since v is unreachable (u, v) must be ineligible, so $yz(u, v) \leq d(u, v) - \Delta/2$ so it satisfy domination after dual adjustment. When $v \in T_{in}$, only approximate tightness can be violated so we can assume $(u, v) \in M$. But since v is unreachable, (u, v) is ineligible, $yz(u, v) \geq d(u, v) - \Delta/2$, so approximate tightness $yz(u, v) \geq d(u, v) - \Delta/2$ holds after dual adjustment.

When both u and v are reachable, if $u \in T_{in}$ and $v \in T_{out}$ (or vice versa), $yz(u, v)$ is unchanged. For the other two cases, (u, v) must be unmatched. If both u and v is in T_{out}, $yz(u, v)$ is incremented by Δ and we only need to worry about domination. But since at the time of dual adjustment there is no reachable blossom or augmenting path, (u, v) cannot be eligible, meaning that $yz(u, v) < d(u, v)$. By Lemma 4.11, this will imply $yz(u, v) \leq d(u, v) - \Delta$ and dual adjustment does not violate domination. When $u, v \in T_{in}$, since $yz(u, v)$ is decremented by Δ, so domination on (u, v) is preserved. This finishes the proof. □

We claim that the search procedure can be implemented in graph metric in $O(m)$ time, without directly computing the distance matrix. We will show this in Section 4.4. With this claim, we are going to present our scaling algorithm for approximate min weight perfect matching.

4.3.2 The Main Algorithm

In this section, we show that for graphs with slightly super linear growth, one can achieve an additive ϵn approximation for the graphic-MWPMproblem. For Christofides algorithm, this suffices to yield a $(3/2 + \epsilon)$-approximate solution in near linear time since the objective value is at least n. Our algorithm uses a scaling technique similar to [9] for bipartite graph. The algorithm consists of L scales. Each scale will run on a subset $T_i \subseteq T$ of terminals and produce a matching M_i (not necessarily perfect) on T_i. Initially $T_1 = T$. At the end of each scale, we commit M_i to the final solution and start the next scale with $T_{i+1} = T_i \setminus V[M_i]$, i.e., matched vertices become non-terminals, but are not removed from the metric. In the last scale, M_L will be a perfect matching on T_L

The goal of each scale is to use Edmonds' Search to *sparsify* the set of unmatched vertices. Specifically, we produce primals and duals M_i, y_i, z_i, Ω_i that satisfy Property 4.7 with some exponentially increasing error parameter Δ_i. Notice that, in Edmonds' Search we work with the *rounded* metric, i.e., every distance is rounded *up* to its nearest multiple of Δ_i, $d_i(u, v) = \Delta_i \lceil d(u, v) \Delta_i \rceil$. Then we have:

Invariant 4.13. *M_i, y_i, z_i, Ω_i satisfy Property 4.7 with respect to distance function d_i with parameter Δ_i.*

The main goal is for T_{i+1} to be sparse in terms of T_i. At each scale, the radius of the dual ball λ_i of a free terminal will increase from D_{i-1} to some target radius R_i.

Property 4.14. *For every $v \in T_i$, $\lambda_i(v) \leq D_i$. At the end of each scale, $\lambda_i(v) = D_i$ if $v \in T_i \setminus V[M_i]$.*

Each scale starts with terminals T_i, and $y_i(u) = {D_{i-1}}^2$ (we define $D_0 = 0$), repeatedly runs Edmonds' Search with granularity Δ_i until y_i for all the free terminals reaches D_i. This will produce a partial matching M_i. We commit the matching M_i to the final solution and the remaining free terminals goes to the next scale.

The final solution will be the union $\widehat{M} = \bigcup_{i=1}^{L} M_i$. To analyze its quality, we define its corresponding duals $\widehat{y}, \widehat{z}, \widehat{\Omega}$ as follows: For $\widehat{y}$ values, let $\Delta y_i(u)$ be the change in $y_i(u)$ during scale i, we define $\widehat{y} = \sum_{i=1}^{L} \Delta y_i$. We take $\widehat{\Omega} = \bigcup_{i=1}^{L} \Omega_i$. Notice that Ω_i and Ω_j are disjoint for every $i < j$. This is because no blossom in Ω_i can contain more than 1 free T_i vertex with respect to M_i, while Ω_j is on $T_j \subseteq T_{i+1}$. This give the natural definition for $\widehat{z}$ values:

$$\widehat{z}(B) = \sum_{i=1}^{L} z_i(B)$$

Notice that at most one term in the summation is positive. The corresponding aggregations of duals is defined similarly:

$$\widehat{yz}(u,v) = \widehat{y}(u) + \widehat{y}(v) + \sum_{B \in \widehat{\Omega} \wedge |B \cap \{u,v\}|=1} \widehat{z}(B)$$

And

$$\widehat{\lambda}(u) = \widehat{y}(u) + \sum_{B \in \widehat{\Omega}: u \in B} \widehat{z}(B)$$

Similarly we have $\widehat{yz}(u,v) \leq \widehat{\lambda(u)} + \widehat{\lambda(v)}$.

Lemma 4.15 (Approximate Domination)**.** *Let (u,v) be a pair such that $u \in T_i \setminus V[M_i]$ and $v \in V[M_j]$ and $i \leq j$. We have*

$$\widehat{yz}(u,v) \leq d(u,v) + e_i + e_j \tag{4.2}$$

[2]This is the value that sums out all the λ_j values from all previous scales j.

where

$$e_i = \begin{cases} \Delta_i & \text{when } i = 1 \\ D_i & \text{when } i > 1 \end{cases}$$

Proof. When $i = j = 1$, by Invariant 4.13 and Property 4.7, we have

$$yz_1(u,v) \leq d_1(u,v) \leq d(u,v) + \Delta_1 < d(u,v) + 2e_1$$

Moreover, we have $\widehat{yz}(u,v) = yz_1(u,v)$. This is because both u and v get matched in the first scale and thus do not participate in any further scale and their y-values remain unchanged. For the same reason, no blossoms in Ω_i for $i > 1$ contain u or v, proving the claim.

When $1 = i < j$, we have $\widehat{\lambda}(u) = \lambda_1(u) \leq yz_1(u,v) \leq d_1(u,v) \leq d(u,v) + \Delta_1$. The first equality is because u only participate in scale 1. $\widehat{\lambda}(v) = \lambda_j(v) \leq D_j$. Adding the two inequalities proves the claim.

The case when $1 < i \leq j$ follows similarly by observing $\widehat{\lambda}(u) = \lambda_i(u) \leq D_i$ and $\widehat{\lambda}(u) = \lambda_j(v) \leq D_j$. □

Lemma 4.16 (Approximate Tightness)**.** *For all $(u,v) \in M_i$, we have $\widehat{yz} \geq d(u,v) - \Delta_i$.*

Proof. Since u and v does not take part in any scale greater than i, we have $\widehat{yz} = yz_i(u,v)$. By Invariant 4.13 and Property 4.7, $\widehat{yz}(u,v) \geq d_i(u,v) - \Delta_i \geq d(u,v) - \Delta_i$. □

Moreover, although we lose laminarity, blossoms in $\widehat{\Omega}$ still retain their structural properties.

Lemma 4.17. *For any $B \in \widehat{\Omega}$, $|\widehat{M} \cap \delta(B)| = 1$.*

Proof. Let $B \in \Omega_i$. No matched edges from below scale i touches B. If B is matched in scale i the claim is trivial. Otherwise its base is the unique vertex in B that is unmatched and goes into T_{i+1} and its incident matched edge will cross B. □

Define $N_i = |V[M_i]|$. We can bound the error term as:

$$
\begin{aligned}
w(M_{opt}) &= \sum_{(u,v)\in M_{opt}} d(u,v) \\
&\geq \sum_{(u,v)\in M_{opt}} \widehat{yz}(u,v) - \sum_{i=1}^{L} |V[M_i]|e_i && \text{(Lemma 4.15)} \\
&\geq \sum_{u} \widehat{y}(u) + \sum_{B\in\widehat{\Omega}} \widehat{z}(B) - \sum_{i=1}^{L} N_i e_i && \text{(definition of } yz\text{)} \\
&\geq \sum_{i=1}^{L} \sum_{(u,v)\in M_i} (\widehat{yz}(u,v)) - \sum_{i=1}^{L} N_i e_i && \text{(Lemma 4.17, } \widehat{z}\geq 0\text{)} \\
&\geq \sum_{i=1}^{L} \sum_{(u,v)\in M_i} (d(u,v) - \Delta_i) - \sum_{i=1}^{L} N_i e_i && \text{(Lemma 4.16)} \\
&= w(\widehat{M}) - \sum_{i=1}^{L} N_i(e_i + \Delta_i/2) && (4.3)
\end{aligned}
$$

We can now prove the main theorem:

Theorem 4.5. *Let $G = (V, E)$ be an unweighted and undirected graph that has n vertices and m edges, and growth $f(d) \geq d\log^2 d$. Let $T \subseteq V$ be a set of terminals and $d : T \times T \mapsto \mathbb{Z}_{\geq 0}$ be the distance metric on G restricted to T. Then an ϵn additive approximate solution to the graphic-*MWPM *problem can be computed in time $O(m \log n 2^{O(1/\epsilon)})$ time. For a graph with polynomial growth, i.e., $f(d) \geq d^{1+\tau}$, an ϵn additive approximate solution can be obtained in time $O(m \log n(1/\epsilon)^{1/\tau})$.*

Proof. We set $\Delta_1 = 1/\lceil 1/\epsilon \rceil$ and $\Delta_i = 2\Delta_{i-1}$ for $1 < i \leq L$. For $0 < \epsilon < 1$, $\epsilon/2 < \Delta_0 < \epsilon$. We also set $D_1 = \lceil 2^{1/\epsilon+1} \rceil + \Delta_1$ and $D_i = 2D_{i-1}$. For the algorithm to find a perfect matching in the last scale, the radius D_L should satisfy $D_L > n/2$, i.e. $L < \lceil |\log(n/2)/(2^{1/\epsilon})| \rceil \leq \log n$.

The growth assumption allows us to bound the density of free terminals at each scale:

Lemma 4.18. *For $i > 1$, $N_i \leq n/(2^{1/\epsilon+i-1}(1/\epsilon + i - 1)^2)$.*

Proof. By Property 4.14, at the end of scale $i - 1$, we have for every pair free of vertices u and v: $d(u,v) \geq d_i(u,v) - \Delta_i = \lambda_i(u) + \lambda_i(v) - \Delta_i = 2D_i - \Delta_i > 2^{1/\epsilon+i}$.

Hence the balls $B(u, 2^{1/\epsilon+i-1})$ and $B(v, 2^{1/\epsilon+i-1})$ must be disjoint, so we have:

$$\begin{aligned} n &\geq \left| \bigcup_{u \in T_i} B(u, 2^{1/\epsilon+i-1}) \right| \\ &= \sum_{u \in T_i} B(u, 2^{1/\epsilon+i-1}) \\ &\geq N_i (2^{1/\epsilon+i-1}) \log^2 (2^{1/\epsilon+i-1}) \end{aligned}$$

Hence $N_i \leq n/(2^{1/\epsilon+i-1}(1/\epsilon + i - 1)^2)$. □

Then the error term $\sum_{i=1}^{L} N_i(e_i + \Delta_i/2)$ in (4.3) can be bounded by:

$$\begin{aligned} \sum_{i=1}^{L} N_i(e_i + \Delta_i/2) &= N_1 \frac{3}{2}\epsilon + \sum_{i=2}^{L} N_i(D_i + \Delta_i) \\ &\leq \frac{3}{2}\epsilon n + \sum_{i=2}^{L} \frac{n}{2^{1/\epsilon+i-1}(1/\epsilon + i - 1)^2} (2^{i-1}\lceil 2^{1/\epsilon} \rceil + 2^i \epsilon) \\ &\leq \frac{3}{2}\epsilon n + \sum_{i=2}^{L} \frac{n}{2^{1/\epsilon+i-1}(1/\epsilon + i - 1)^2} (1+\epsilon)(2^{1/\epsilon+i}) \\ &= \frac{3}{2}\epsilon n + 4(1+\epsilon) \sum_{i=1}^{L} \frac{n}{(1/\epsilon + i)^2} \\ &\leq \frac{3}{2}\epsilon n + 5\epsilon n \end{aligned}$$

The fourth line is by $\lceil 2^{1/\epsilon} \rceil \leq 2^{1/\epsilon+1}$ and $2^{1/\epsilon} + \epsilon \leq (1+\epsilon)2^{1/\epsilon}$ for $0 < \epsilon < 1$. And the last line is by $\sum_{i=1}^{\infty} \frac{1}{(\epsilon+i)^2} \leq \int_{1/\epsilon}^{\infty} \frac{1}{x^2}\, dx = \epsilon$. Hence setting $\epsilon = 2/13\epsilon'$ we get an algorithm of additive error $\epsilon' n$.

Time complexity We interchange the notation and let ϵ be the additive approximation ratio in ϵn we are looking for and let $\epsilon' = 2/13\epsilon$ as in the proof. At each scale the number of iteration in Edmonds' Search is $D_i/\Delta_i = O(2^{1/\epsilon'}/\epsilon') = 2^{O(1/\epsilon)}$. Hence one scale can be implemented in $O(m2^{O(1/\epsilon)})$ time. Since there are a total of $O(\log n)$ scales, the total running time is $O(m \log n 2^{O(1/\epsilon)})$ □

With similar technique, if we allow the growth to be *polynomially* super linear, say the growth function is $f(d) = d^{1+\tau}$ for some constant $\tau > 0$, we can achieve running time that is polynomial in $1/\epsilon$. This can be done by changing the initial radius to $D_1 = \lceil (1/\epsilon)^{1/\tau} \rceil$. This gives the following theorem:

Theorem 4.19. *Let $G = (V, E)$ be an unweighted and undirected graph that has n vertices and m edges, and growth $f(d) = d^{1+\tau}$ for some constant $\tau > 0$. Let $T \subseteq V$ be a set of terminals and $d_G : T \times T \mapsto \mathbb{Z}_{\geq 0}$ be the distance metric on G. Then an ϵn additive approximation to a min weight perfect matching on metric d_G can be computed in time $O(m \log n(1/\epsilon)^{1/\tau})$ time.*

4.4 Implementing Edmonds' Search in Graph Metrics

In this section we introduce a near linear time implementation of Edmonds' Search. The goal is to implement each step of Edmonds' Search directly on the original graph G without computing the distance matrix among terminals and explicitly constructing of the metric graph $\mathcal{G}$. This avoids having to compute an instance of all pairs shortest paths, which is too costly when we look for a near linear time algorithm.

Granularity and Rounded Metric Throughout this section we will assume there exists some constant Δ_0 such that all granularity parameters Δ in the Edmonds' Search are multiples of Δ_0. Moreover, $\Delta_0/2$ must be a divisor of 1. To carry out dual adjustment by Δ when Δ is a small integral divisor of 1, we will first subdivide all edges in G into paths of length $2/\Delta_0$, such that a unit of dual adjustment by $\Delta_0/2$ can be represented by growing or shrinking BFS trees $\mathcal{T}$ by one layer. In this section, we use n and m to denote the number of vertices and edges in the subdivided graph.

However when $\Delta > 1$, the distance in the original graph metric d might not always be a multiple of Δ, as assumed in Lemma 4.9. This can invalidate the correctness of Edmonds' Search. To circumvent the issue, for a fixed Δ, we define the *rounded* metric d' on the same vertex set as $d'(u, v) = \Delta\lceil d(u, v)/\Delta \rceil$. For the rounded metric the granularity assumption holds. We will show that we can *simulate* Edmonds' Search on rounded metrics with a *truncated* Edmonds' Search on the original metric.

Data Structures Throughout the entire process of Edmonds' Search we maintain all dual values x and y, as well as a blossom tree representing Ω. We also maintain all dual balls for each outermost blossom, using a collection of *truncated* breadth first search trees rooted at each vertex of blossom as discussed in Section 4.2.2. The BFS tree is truncated in the sense that we also enforce the following invariant:

Invariant 4.20. *Two distinct BFS trees only intersect at their leaves.*

Therefore, a BFS tree might only contain a subset of vertices in its corresponding dual ball. Nevertheless, we can show that the truncated BFS tree still possess enough information to recover the eligibility subgraph $\mathcal{G}_{\text{elig}}$. This is captured in the Invariant 4.21.

Invariant 4.21. *If u and v are two vertices not in the same blossom and $d'(u,v) = \lambda(u) + \lambda(v)$, the BFS trees $\mathcal{T}_u$ and $\mathcal{T}_v$ must intersect at some vertex. Conversely, if BFS trees $\mathcal{T}_u$ and $\mathcal{T}_v$ intersect at some vertex, we must have $d'(u,v) = \lambda(u) + \lambda(v)$.*

We say an outermost blossom B *owns* a vertex v in G, terminal or nonterminal, if v is a vertex in the tree $\mathcal{T}_u$ for some terminal $u \in B$. Invariant 4.20 implies that for any vertex v in G, it has at most $\deg(v)+1$ owners. This is because if v has one than 1 owner, for each edge (v,w) such that v and w share an owner, w must be a nonleaf at its corresponding BFS tree and thus has exactly 1 owner. Hence we can use a linked-list at each vertex to keep track of its set of owners. Moreover, Invariant 4.21 implies that if a vertex w in G has owners u_1 and u_2 such that (u_1, u_2) is unmatched and u_1, u_2 are in distinct outermost blossoms, the edge (u_1, u_2) must be eligible. The converse is also true.

In the end, we also keep track of the matching pairs in M.

Augmentation. The implementation is based on the depth first search algorithm in [75]. We will state their algorithm and its invariant but due to limitation of space we will refer the reader to [75] for the proof of correctness. We will then describe how to simulate the procedure on the original graph without explicitly construct $\mathcal{G}$.

The input to the algorithm is the contracted graph $\overline{\mathcal{G}_{\text{elig}}}$. The algorithm maintains a depth first search forest $\mathfrak{F} = \langle \mathfrak{T}_1, \mathfrak{T}_2, \dots, \mathfrak{T}_k \rangle$ rooted at free vertices. Each search tree is grown in a greedy and depth first fashion to find all vertices reachable from its root via an alternating path.

Similar to depth first search, $\mathfrak{T}_1, \mathfrak{T}_2, \dots, \mathfrak{T}_{k-1}$ are search trees from previously terminated searches, and $\mathfrak{T}_k$ is the *active* tree. The active tree is grown in a depth first manner, i.e., its "rightmost" path consists of vertices currently in the DFS stack. The path is always an even length alternating path that starts at the root and ends at the current vertex we are examining. Each node in the graph has one of 3 statuses: *Unvisited*, meaning the node is not visited by the search algorithm; *Inner/outer*, meaning the node has been visited by the search algorithm and labelled as *Inner/Outer*.

Each terminated search terminates by either discovering an augmenting path, or concluding no augmenting path exists from its root to a free node that has not been

searched yet. We use Ψ to denote the set of augmenting paths discovered by the algorithm.

The algorithm also maintains a set of nested blossoms Ω' on $\overline{\mathcal{G}_{\text{elig}}}$ discovered during the search.

A search start at a free node as the root of the search tree and the only node in the active path. This node marked *outer* node. Suppose u is the last (outer) node on the active path. We *scan* all the unmatched edges (u, v) incident to u and depending on the status of v, we do one of the following:

1. *Augmentation.* If v is unvisited and unmatched, extend the active path to v and conclude the active path form an augmenting path. Mark v as outer and terminate the search.
2. *DFS Extension.* If v is unvisited and matched to v', we extend the active path and active tree with edge (u, v) and (v, v'), where v is the parent of v' and u is the parent of v. We label v as inner and v' as outer. Now v' is the last node on the active path and we continue the search from v'.
3. *Blossom Formation.* If v is visited and is an outer descendant of u in the active tree $\mathfrak{T}_k$. Let $P = \langle u = v_1, v_2, \ldots, v_l = v \rangle$ be the sequence of vertices or shrunken blossoms on the tree path from u to v. Extend the active path from u to v_2 by going from u to v, then back in ancestral direction up to v_2. Notice that all the nodes with even indexes are originally inner. Label them now as outer. Now v_2 is the last node in the active tree. Finally, form and contract a blossom B consisting of all the nodes in P and add it to Ω'. Notice that inner nodes on the path are all singletons and outer nodes can be singleton or blossoms. Data structures such as [73] allows us to keep track of all the blossoms in a total of linear time.
4. *DFS retraction.* If all unmatched edges (u, v) are explored without finding an augmenting path, we retract from u by removing u and its parent from the active path. Its grandparent, which must be outer, is now the last vertex on the active path. If u is the root of the tree terminate the search.

Notice that we ignore the edge (u, v) when v is an inner node, or when v is an outer ancestor of u. The process maintains the following properties:

Lemma 4.22. *Suppose u and v are outer vertices and (u, v) is scanned from both of its endpoints. Then (u, v) must be in the same blossom in Ω'.*

The search terminates by performing a search from all unvisited free nodes. When the algorithm terminates, each free node is either a root of a terminated search tree or involve in some augmenting path in Ψ.

The correctness of the algorithm is stated in [75].

Lemma 4.23 ([75] Lemma 8.1)**.** *When the algorithm terminate, Ψ is a maximal set of vertex disjoint augmenting path.*

Now we discuss its implementation on graph metrics. The main issue we need to handle is that a nonterminal with s owners can generate $\binom{s}{2}$ edges in $\mathcal{G}_{\text{elig}}$, and constructing them explicitly can lead to $\Omega(n^2)$ running time regardless of the density of G. To resolve the issue, each nonterminal keeps track of all its owners along with their inner/outer status. By previous discussion, for each edge (u, v) in the original graph, u and v shares at most one owner, we refers to a specific owner of u by one of its neighboring edges.

Notice that the status of a node in $\overline{\mathcal{G}_{\text{elig}}}$ can only transition from unvisited to inner, unvisited to outer, and inner to outer. Moreover a node's transition from inner to outer only occurs at the blossom formation step. Hence for each nonterminal v, we keep track of 3 sets, $S_v^{(U)}$, the set of unvisited owners, $S_v^{(I)}$, the set of inner owners, and $S_v^{(O)}$, the set of outer owners.

For DFS extension, the current outer vertex u visits all the nonterminals v it owns, finds one of its unvisited owners $w \in S_v^{(U)}$ and continues the search by extending to w and w' where w' is matched to w'. Doing so will move w from the unvisited set to inner set and w' to the outer set. The cost is charged to moving w and w' to their respective sets.

Similarly for augmentation, when the node w is unmatched, we terminate the search and simply remove it from the unvisited set since it no longer participates in any future search.

For blossom formation, the current node u finds an nonterminal v that it owns and a outer node w in $S_v^{(O)}$. If w is u's descendant, we perform a blossom formation step and convert all inner nodes in the tree path from u to w to outer. This step will not change the status of w, but it will put u and w into the same outermost blossom, which can be simulated by a concatenating two linked lists in $S_v^{(O)}$ that each containing u or w into a single linked list. The cost for inspecting w when w is an ancestor of u can be charged to the cost when we inspect u from w later in the search. Notice that this search must happen because an outer node will always remain outer.

At last, implementing DFS retraction is trivially done by retracting from the DFS tree after exhausting all the unvisited sets and outer sets.

This leads to the following lemma:

Lemma 4.24. *A maximal set of augmenting path in $\mathcal{G}_{\text{elig}}$ can be found in time $O((n+m))$ time where n and m are the number of vertices and edges in the G.*

Proof. By Invariant 4.20, the total number of owners of a nonterminal v is at most $\deg(v) + 1$. Hence the total number of (owner, ownee) pairs, i.e. the total size of $S_v^{(U)}$ at the start of the algorithm is $O(m + n)$. Because we explicitly keep track of the set of unvisited and outer owners, no exploration will be done from the current node to the inner node. Each exploration to unvisited node will be charged to the removal of the node from the unvisited set, and each exploration to an outer node will be charged to the concatenation operation that merges the two owners. Hence each operation can be done in constant time and we can perform all operations in $O(n + m)$ time. □

Blossom Formation. Blossom formation can also be implemented by the augmentation algorithm above. By Lemma 4.22, Ω' forms a maximal set of nested blossom on $\mathcal{G}_{\text{elig}}$. Therefore, we can simply run the algorithm for augmentation (which will not return any augmenting path), and use Ω' as a maximal set of reachable blossom.

Dual Adjustment. Dual adjustment can be simulated by extending or retracting the breadth first search trees. Specifically, if vertex u is reachable, each dual adjustment either increments or decrements $\lambda(u)$ by $\Delta/2$. We simulate this change by Δ/Δ_0 extensions or retractions of breadth first search tree, i.e. growing or shrinking the depth first search tree by Δ/Δ_0 layers. After the entire process, the distance from root and the outermost layers of the tree increases or decreases by $\Delta/2$.

However since the distance function might not always be multiple of Δ, growing breadth first search tree by Δ/Δ_0 might violate Invariant 4.20. To enforce Invariant 4.20, when growing a tree, a vertex v will stop searching for its children once its layer is at the target distance of the dual adjustment, or v is in more than 1 breadth first search tree.

Shrinking is more straightforward, we shrink the outermost layer of breadth first search tree $\mathcal{T}_u$ until its outermost layer is at the distance $\lambda(u)$ after the dual adjustment. No violation of Invariant 4.20 or Invariant 4.21 can be caused in this stage.

The total running time for implementing one dual adjustment, regardless of the magnitude of Δ, is linear in terms of the size of the graph after subdivisions of edges. This is because each edge can be visited at most twice, one from shrinking a tree, one from growing the tree. As a result, the total running time is $O(n + m)$.

Lemma 4.25. *One step of dual adjustment can be implemented in $O(n+m)$ time.*

Blossom Dissolution Blossom Dissolution does not change the dual balls or the breadth first search tree at a terminal level. The changes happen only at the blossom tree and can be easily implemented in linear time.

Corollary 4.26. *One iteration of Edmonds' Search can be implemented in $O(m)$ time.*

4.5 Exact Algorithm for Bounded Treewidth Graph

In this section, we prove that if we are given a tree decomposition of width k, we can solve the graphic-MWPMproblem in $O(mk\log^2 n)$ time. The high level idea is straightforward: we use the tree decomposition to construct a hierarchy of small balanced separator. This allows us to apply the divide and conquer algorithm from[16] and [136] using the balanced separator as portals.

4.5.1 Treewidth and Hierarchical Separators

We first prove the fact that we can obtain a hierarchy of balanced separator from a tree decomposition.

Definition 4.27. *Given $G=(V,E)$, a set of vertices $S\subset V$ is a separator if $G-S$ contains more than 1 connected component.*

Definition 4.28. *Given $G=(V,E)$, and $\alpha: V\mapsto\{0,1\}$, a separator S is balanced if every component V_i in $G-S$ has $\alpha(V_i)$ at most $(2/3)\alpha(V)$.*

Lemma 4.29 (Separator for tree decomposition)**.** *Let $(\mathcal{T},\{B_x\}_{x\in V(\mathcal{T})})$ be a tree decomposition for G. Let $(x,y)\subset E(T)$ be an edge in $\mathcal{T}$. Then $B_x\cap B_y$ is a separator for G.*

Proof. Let T_x and T_y denote the set of vertices in the connected component of $\mathcal{T}-\{(x,y)\}$ that contains x and y respectively. Let $V_x,V_y\subseteq V(G)$ denote the union of their corresponding bags. We show that for every edge $(u,v)\in E(G)$ such that $u\in V_x$, $v\in V_y$, either u or v or both belongs to $B_x\cap B_y$. This proves that $B_x\cap B_y$ is a separator.

By definition we know there exists $w\in V(\mathcal{T})$ such that B_w contains both u and v. Without lost of generality assume $w\in T_x$. Notice that there is also $v'\in T_y$ that

$B_{v'}$ contains v. Since $\mathcal{T}[v]$ is connected v must be contained in the bags on the $\mathcal{T}$ path from w to u', in particular in both B_x and B_y. □

Corollary 4.30. *Any bag B_x is a separator.*

Lemma 4.31. *Let $G = (V, E)$ be a graph of treewidth k, and $\alpha : V \mapsto \{0, 1\}$. G has a balanced separator of size at most $k + 1$. Moreover, given a tree decomposition $(\mathcal{T}, \{B_x\}_{x \in V(\mathcal{T})})$ of width k, one can find such a separator in linear time.*

Proof. Select an arbitrary node r of $\mathcal{T}$ as root. Let $\mathcal{T}_x$ denote the subtree rooted at x. Let $\alpha(\mathcal{T}_x) = \sum_{u \in \bigcup_{y:y \in V(\mathcal{T}_x)}} \alpha(u)$. Using linear time, we can find a node w (possibly r) such that:

1. $\alpha(\mathcal{T}_w) \geq (2/3)\alpha(\mathcal{T})$
2. Any child w' of w has $\alpha(\mathcal{T}'_w) \leq (2/3)\alpha(\mathcal{T})$.

Then B_w is a balanced separator. □

Corollary 4.32. *Given a graph G of treewidth $k = o(n)$, we can in linear time decompose its vertex set into two components V_1, V_2 and an interface set I such that:*

1. *$V_1 \cup V_2 = V$.*
2. *$\emptyset \neq I \subset V_1 \cap V_2$.*
3. *For every pair of vertices u_1, u_2 such that $u_1 \in V_1$ and $u_2 \in V_2$, every path between u_1 and u_2 contain a vertex in I.*
4. *$|V_1|, |V_2| \leq (2/3 + o(1))n$.*

Moreover, V_1 and V_2 can be obtained in linear time.

Proof. Let S be the separator guaranteed by Lemma 4.31, and $C_1, C_2, \ldots, C_k$ be connected components in $G - S$. If there is a component, say C_1, have size $(1/3)n \leq |C_1| \leq (2/3)n$, can set $V_1 = C_1 \cup S$ and $V_2 = (C_2 \cup C_3 \cup \ldots \cup C_k) \cup S$ and we are done. Otherwise, all C_i has size at most $(1/3)n$. Take $1 \leq p < k$ be the maximum number such that $|C_1 \cup C_2 \cup \ldots \cup C_p| \leq (2/3)n$. We have $|C_{p+1} \cup C_{p+2} \cup \ldots \cup C_l| = |C_{p+1}| + |C_{p+2} \cup C_{p+3} \cup \ldots \cup C_k| \leq (1/3)n + (1/3)n = (2/3)n$. Setting $V_1 = (C_1 \cup C_2 \cup \ldots \cup C_p) \cup S$ and $V_2 = (C_p \cup C_{p+1} \cup \ldots \cup C_k) \cup S$ guarantees the bound. The rest of the properties follows from the construction and Lemma 4.31. This procedure takes linear time. □

Corollary 4.32 provides a recursive decomposition scheme that decomposes a graph G of treewidth k into a hierarchy of components connected by a small vertex cut. The hierarchy consists of $L \leq \log_{3/2} n$ *levels*, we use $\mathcal{H} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_L\}$ to denote the

class of components at each level. The first level $\mathcal{K}_1$ consists of a single component, the whole graph G. For level $i \geq 2$, suppose $\mathcal{K}_{i-1} = \{G_1, G_2, \ldots, G_t\}$, then the level i components in $\mathcal{K}_i$ is obtained by applying Corollary 4.32 to each $G_j \in \mathcal{K}_{i-1}$. The procedure splits each G_j into exactly two induced subgraphs that intersect at most $k+1$ vertices. We call these vertices *portals*, as in Arora's algorithm. Let P_i be the set of portals introduced when splitting components in $\mathcal{K}_{i-1}$ to $\mathcal{K}_i$ and define $P_1 = \emptyset$. For any components $H \in \mathcal{K}_i$, the portals of H, $\rho(H)$ is the set $(P_1 \cup P_2 \cup \ldots \cup P_i) \cap V(H)$ and let P denote the set of all portals. The following bound follows immediately from construction.

Lemma 4.33. *For any component H in the hierarchy, $|\rho(H)| \leq (k+1)\log n$.*

Notice that the set of components in the hierarchy $\mathcal{H}$ form a complete binary tree with root G. Components in $\mathcal{K}_i$ are nodes at depth i and $\mathcal{K}_L$ are the set of leaves.

4.5.2 The Divide-and-Conquer Framework for MWPM

In Varadarajan and Agarwal's paper[136], they present an divide and conquer framework for plane min weight perfect matching [136]. In this section, we review their framework and describe how to apply the framework in graph metrics with low treewidth.

Recall that in the graphic-MWPMproblem, we are given an unweighted undirected graph G and a set of terminal T. Suppose G has treewidth k and we have a hierarchical decomposition specified as above. We now define the *metric graph* $\mathcal{G}$ as the metric completion on the set of terminals T *and portals* P. We first construct a new weighted graph $\mathbb{G}$ as follows: The vertex set of $\mathbb{G}$ is the set of terminals T and the set of all portals P. First we clean up the instance such that each terminal belongs to exactly one leaf component. This can be done by assigning the terminal to an arbitrary leaf component when it is shared by two leaf components (when the terminal is also a portal in the decomposition). Then for each leaf component H and each terminal u in H, add an edge between u and each portal $p \in \rho(H)$ of weight $d_G(u,p)$. Moreover, for each pair of portals $p, q \in \rho(H)$, add an edge (p,q) of weight $d_G(p,q)$. The total number of vertices is $O(n)$. Each terminal has degree at most $k+1$, and each portal has degree at most $2k$ (each portal is shared by at most 2 leaf components), so the total number of edges is $O(kn)$.

The algorithm maintains a matching $M \subset T \times T$, a laminar set Ω of blossoms and dual functions y, z and their aggregation λ that satisfy the following:

Property 4.34 (Exact Complementary Slackness)**.** *Let M be a matching on $T \times T$ and y, z, λ be the dual functions and Ω be the set of blossoms:*

1. *Domination. For all $(u, v) \in T \times T$, $yz(u, v) \leq d(u, v)$.*
2. *Tightness. For all $(u, v) \in M \cup (\bigcup_{B \in \Omega} E_B)$, $yz(u, v) = d(u, v)$.*
3. *Nonnegative Dual. For each blossom B in Ω, $z(B) \geq 0$.*
4. *Structural Blossoms. Ω forms a laminar family of blossom and for each blossom $B \in \Omega$, the matching M is maximal inside B, i.e., $|M \cap E(B)| = (|B| - 1)/2$.*

Fact 4.35. *Suppose M is a perfect matching along with duals y, z and Ω that satisfies Property 4.34. Then M is a min weight perfect matching.*

The property was introduced by Edmonds [59] and used in a series of subsequent works[76, 70]. Following this line of work, the set of *eligible* pairs is defined as the set of pairs $(u, v) \in T \times T$ such that $yz(u, v) = d(u, v)$. The eligible subgraph $\mathcal{G}_{\text{elig}}$ is defined similarly as in Section 4.2. The main implication of such definition is that the version of Edmonds' Search here no longer performs dual adjustment in a uniform rate as in Section 4.4. Here we use a priority queue to keep track of the minimum amount of dual adjustment needed to trigger a set of combinatorial events, e.g. discovery of an augmenting path or a blossom consists of eligible edges and dissolution of a blossom. The progress of the algorithm is now measured by the number free vertices remaining, rather than their λ-values.

Before we describe the priority queue data structure and the algorithm, we need to introduce some additional property that the partial primal and dual solution need to satisfy in order for the algorithm to efficiently take advantage of the hierarchical decomposition $\mathcal{H}$. Following the work of [136], we use a divide-and-conquer scheme to compute the matching. This requires the dual values to satisfy one additional property:

Property 4.36 (Portal Constraint)**.** *Let H be a component in hierarchy $\mathcal{H}$, and let M, Ω, y, z be defined in Property 4.34. We require that for all pairs of (u, p) where $u \in T$ and $p \in \rho(H)$ that $\lambda(u) \leq d(u, p)$.*

This property guarantees that once recursive calls for children components finishes, the partial solution for both children components still combine to satisfy Property 4.34 and Property 4.36 for their parent component. We say an unmatched terminal u is *tied to* a portal p if $\lambda(u) = d(u, p)$. The recursive call for a component H returns with a matching M and duals y, z, Ω satisfying the following invariant:

Invariant 4.37. *The matching M, and duals y, z, Ω satisfy the Property 4.34 and Property 4.36 for component H. Moreover, each terminal in H is either matched or tied to a portal in $\rho(H)$. Moreover, at any moment, no two terminals are tied to the same portal.*

For any portal p, we use $c(p)$ to denote the terminal that is tied to p.

Now we state the recursive procedure, call it EDMONDS(H). It first recursively calls EDMONDS(H_1) and EDMONDS(H_2) for its children H_1 and H_2. Let $(M_1, y_1, z_1, \Omega_1)$ and $(M_2, y_2, z_2, \Omega_2)$ be the partial solution returned. The procedure starts with the initial solution $(M_0, y_0, z_0, \Omega_0)$ obtained by taking $M = M_1 \cup M_2$, $\Omega = \Omega_1 \cup \Omega_2$ and y, z being the function taking the disjoint union of y_1, y_2 and z_1, z_2 respectively. If H is a leaf component, then $M = \emptyset$, $\Omega = \emptyset$, $y(u) = 0$ and $z(B) = 0$.

Before we state the iterative search procedure, we define several terminologies first. At any moment, terminals are either matched, free, or *tied*, meaning that it is tied to a portal. A tied terminal is *not* considered free. A blossom is tied if its base is tied. We use F and C to denote the sets of free and tied terminals. A portal is *tied* if there is a terminal tied to it and *free* otherwise. An *augmenting path* is an odd length alternating path from a free terminal to a free terminal, a tied terminal, or a free portal. We call them T to T, T to C and T to P augmenting path respectively.

We run the following procedures until there are no more free terminals:

1. *Augmentation.* While there is a eligible T to T or T to C augmenting path, find a type-T augmenting path P in $\mathcal{G}_{\text{elig}}$ and augment $M \leftarrow M \oplus P$. Update $\mathcal{G}_{\text{elig}}$.
2. *Portal Binding.* While there is a T to P eligible augmenting path, find a type-P augmenting path P. Let (u, p) be the edge on P incident to the portal end p. Set $M \leftarrow M \oplus (P \setminus \{u, p\})$. Set $c(p) = u$.
3. *Blossom Formation.* Let $\overline{T_{out}}$ be the set of nodes reachable from a free node via an even length alternating path. Find a *maximal* set of nested blossoms Ω' on $\overline{T_{out}}$. The maximality here means after contracting blossoms in Ω', the contracted graph does not contain any blossom reachable via an eligible alternating path. Update $\Omega \leftarrow \Omega \cup \Omega'$. Set $z(B) \leftarrow 0$ for any blossom in Ω'. $\mathcal{G}_{\text{elig}}$ does not change in this step.
4. *Dual Adjustment.* Find the set of $\overline{T_{out}}$ and let $\overline{T}_{in}$ be the set of nodes not in $\overline{T}_{out}$ reachable from a free vertex via an odd length alternating path. Let T_{out} and T_{in} be the set terminals in T that is contained in some nodes in T_{out} and T_{in} respectively. The set of remaining terminals that are not in C are labelled

as unreachable, denoted by T_{un}. let Ω_{out} and Ω_{in} be the set of blossoms in $\overline{T}_{out}$ and $\overline{T}_{in}$.

$$\begin{aligned}
\delta_1 &= \min_{B:B\in\Omega_{in}} z(B) \\
\delta_2 &= \min_{u,v:u\in T_{out},v\in T_{un}} (d(u,v) - yz(u,v)) \\
\delta_3 &= \min_{u,v:u,v\in T_{out},B(u)\neq B(v)} (d(u,v) - yz(u,v))/2 \\
\delta_4 &= \min_{u,v:u\in T_{out},v\in C} (d(u,v) - yz(u,v)) \\
\delta_5 &= \min_{u,p:u\in T_{out},p\in P} (d(u,p) - \lambda(u))
\end{aligned}$$

Set $\delta = \min\{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5\}$. Update the dual variable as follows:

$$\begin{aligned}
y(u) &\leftarrow y(u) + \delta \quad \text{if } u \text{ is in } T_{out} \text{ but not in any blossom.} \\
z(B) &\leftarrow z(B) + \delta \quad \text{if } B \text{ is outermost blossom in } T_{out}. \\
y(u) &\leftarrow y(u) - \delta \quad \text{if } u \text{ in } T_{in} \text{ but not in any blossom.} \\
z(B) &\leftarrow z(B) - \delta \quad \text{if } B \text{ is outermost blossom in } T_{in}.
\end{aligned}$$

5. *Blossom Dissolution.* After dual adjustment some outermost blossoms might have 0 z-values. Dissolve those blossom as long as they exists. Update Ω and $\mathcal{G}_{\text{elig}}$.

Notice that if we maintain Property 4.34 and Property 4.36, we have $\delta \geq 0$. And in the event of $\delta = \delta_i < +\infty$, performing a dual adjustment by δ makes it possible to perform one of augmentation, portal binding, blossom formation, or blossom dissolution. Specifically, when $\delta = \delta_1$, an outermost blossom in T_{in} has 0 z-value, in which case we will perform a blossom dissolution. When $\delta = \delta_2$ or $\delta = \delta_4$, dual adjustment creates a new eligible edge (u, v) between an outer terminal and unmatched terminal, which creates a type-T augmenting path. When $\delta = \delta_3$, an edge between two outer terminals become eligible, meaning that we have either discover an augmenting path or we have discovered a new outer blossom.

To bound the total running time, we divide the algorithm into *phases*. Each phase ends when an augmentation or a portal binding operation is performed. In both cases the number of free terminals decreases by at least 1.

Lemma 4.38. *In the initial solution $(M_0, \Omega_0, y_0, z_0)$, the number of free terminals is at most $2(k+1)$.*

Proof. By Invariant 4.37, in the partial solution returned by the recursive call, every terminal is either matched or tied to a portal. Then every matched terminal will still be matched in M_0, the set of terminals stop being tied is the set of terminal in H_1

and H_2 that is tied to the cut set portal shared by H_1 and H_2. By construction, the cut set has size at most $k+1$, so the set of free terminal has size at most $2(k+1)$. □

Observe that if two terminals u_1, u_2 from H_1 and H_2, respectively, is tied to the same portal, one can add (u, v) to the matching and still satisfy Property 4.36, and the bound can be improved to $k+1$.

Corollary 4.39. *The number of phases in each* EDMONDS(H) *is at most* $k+1$.

To efficiently implement one phase, we need a priority queue to quickly keep track of $\{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5\}$ within a phase, especially when blossoms form and dissolve. The following theorem can be obtained by adapting the construction in Appendix 5 in [56], which states that for a weighted graph G with n vertices and m edges, one phase of EDMONDS can be implemented in $O(mp)$ time, where p is the amortized time for a priority queue that supports insert and delete-min.

Theorem 4.40. *Given a priority queue data structure that supports INSERT and delete-min in amortized time* $O(q)$*, one can implement one* phase *in* EDMONDS *in* $O(mq)$ *time.*

The implementation of this data structure can be adapted to both accommodate portals constraint as well as the implicit representation of graph G. The main idea is as follows:

δ_1 can be maintained explicitly with $O(n_H)$ variables. Maintaining $\delta_2, \delta_3, \delta_4$ essentially reduces to maintaining for each pair of terminals u and v, when does their dual ball start touching or stop touching each other. Recall that we say a dual ball $\mathcal{B}_u$ *owns* a portal if it contains it and u is an *owner* of p. Then we need to answer the following two questions:

1. If a portal p does not have an owner, after how many units of dual adjustment does it start to have an owner?
2. If a portal p currently have an owner u, after how many units of dual adjustment will u stop being the owner of p?

To answer the first question, notice that only dual balls of outer terminals are growing, and once a terminal becomes outer it will never become inner or unreachable until the next augmentation. Hence a portal, once it gains an owner, the owner will not change within a phase. Therefore, we only need to keep track of $O(m_H)$ events in the following form: For an edge (p_1, p_2) where p_1 has an owner and p_2 does not have an owner, when will the dual ball containing p_1 touch p_2.

To answer the second question one just needs to use a priority queue with $O(n_H)$ events corresponding to each (owner, ownee) pair such that the ownee is *not* on the boundary of the owner's dual ball. There are a total of n_H events.

Hence the total running time of EDMONDS(H) is $O(km_Hp)$. Summing up all the recursive calls on the same level, since $|V(\mathbb{G})| = O(n)$ and $|E(\mathbb{G})| = O(kn)$, the conquer step at each level takes $O(mkp \log n)$. The total running time of the algorithm is $O(mkp \log n)$, when the tree decomposition is given.

Theorem 4.6. *Given an unweighted and undirected graph $G = (V, E)$ along with a tree decomposition of width k, and a set of terminals $T \subseteq V$, we can solve the graphic-*MWPM *problem in time $O(mkp \log n)$, where p is the update amortized time for a priority queue that supports insert and delete-min.*

CHAPTER 5

Join

5.1 Introduction

In this chapter, we study the effectiveness as well as limitation of sampling in approximating join queries. Sampling is one of the most widely-used techniques for general-purpose AQP [43]. The high level idea is to execute the query on a small sample of the original table(s) to provide a fast, but approximate, answer. While effective for simple aggregates, using samples for join queries has long remained an open problem [8]. There are two main approaches to AQP: *offline* or *online*. Offline approaches [6, 8, 11, 32, 77, 119] build samples (or other synopses) prior to query arrival. At run time, they simply choose appropriate samples that can yield the best accuracy/performance for each incoming query. Online approaches, on the other hand, such as wander-join perform much of their sampling at run time based on the query at hand [18, 41, 86, 93, 118, 140]. Naturally, offline sampling leads to significantly higher speedup, while online techniques can support a much wider class of queries [93]. The same taxonomy applies to join approximation: offline techniques perform joins on previously-prepared samples [8, 30, 107, 119, 35], while online approaches seek to produce a sample of the output of the join at run time [80, 52, 107]. As mentioned, the latter often means more modest speedups (e.g., 2× [93]) which may not be sufficient to justify approximation, or additional requirements (e.g., an index for each join column [107]) which may not be acceptable to many applications. Thus, our focus in this paper—and what is considered an open-problem—is the offline approach: joins on samples, not sampling the join's output.

Joins on Samples The simplest strategy is as follows. Given two large tables T_1 and T_2, create a uniform random sample of each, say S_1 and S_2 respectively, and then use $S_1 \bowtie S_2$ to approximate aggregate statistics of $T_1 \bowtie T_2$. This will lead to

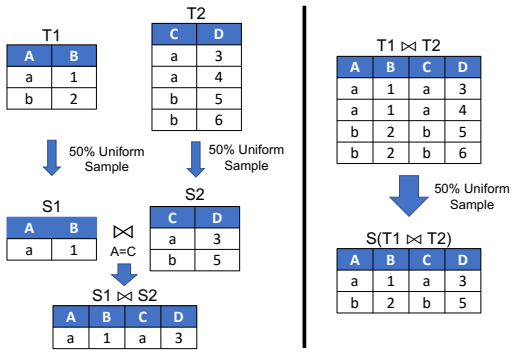


Figure 5.1: A toy example of joining two uniform samples (left) versus a uniform sample of the join (right).

significant speedup if samples are much smaller than original tables, i.e., $|T_i| \gg |S_i|$.

One of the earliest results in this area shows that this simple strategy is futile for two reasons [6]. First, joining two uniform samples leads to quadratically fewer output tuples, i.e., joining two uniform samples that are each p fraction ($0 \leq p < 1$) of the original tables will only produce p^2 of the output tuples of the original join (see Figure 5.1). Second, joining uniform samples of two tables does not yield an independent sample of their join[1] (see Section 5.2.1 for details). The dependence of the output tuples can drastically lower the approximation accuracy [6, 30].

Prior Work *Universe* sampling [81, 93, 119] addresses the first drawback of uniform sampling. Although universe sampling avoids quadratic reduction of output, its creates even more correlation in its output, leading to much lower accuracy (see Section 5.3.1).

Atserias et al. provide a worst case lower bound for any query involving equi-joins on multiple relations, showing that computing exact joins with a small memory or time budget is hard [17]. For instance, the maximum possible join size for any cyclic join on three n-tuple relations is $\Theta(n^{1.5})$. Thus, a natural question is whether approximating joins is also hard with small memory or time.

Our Contribution This paper focuses on understanding the limitation of using offline samples in approximating join queries. Given a sampling budget, how well can we approximate the join of two tables using their offline samples? To answer

[1]Prior work has stated that joining uniform samples is not a *uniform* sample of the join [8]. We avoid this terminology since uniform means equal probability of inclusion, and in this case each tuple does appear in the join of the uniform samples with equal probability, but not independently. In other words, joining two i.i.d. samples is an identical, but not independent, sample of the join.

this question, we must first define what constitutes a "good" approximation of a join. We consider two metrics: (1) output cardinality and (2) aggregation accuracy. The former is the number of tuples of the original join that also appear in the join of the samples, whereas the latter is the error of the aggregates estimated from the sample-based join with respect to their true values, if computed on the original join. Because in this paper we only consider unbiased estimators, we measure approximation error in terms of the variance of our estimators.

For the first metric, we provide a simple proof showing that universe sampling is optimal, i.e. no sampling scheme with the same sampling rate can outperform universe sampling in terms of the (expected) output cardinality. However, as we show in Section 5.3.1, retaining a large number of join tuples does not imply accurate aggregates. It is therefore natural to also ask about the lowest variance that can be achieved given a sampling rate. To the best of our knowledge, this has remained an open problem to date. For the first time, we formally study this problem and offer an information-theoretical lower bound to this question. We also present a hybrid sampling scheme that matches this lower bound within a constant factor. This scheme involves a centralized computation, which can become prohibitive for large tables due to large amounts of statistics that need to be shuffled across the network. Thus, we also propose a decentralized variant that only shuffles a minimal amount of information across the nodes—such as the table size and maximum frequency—but still achieves the same worst case guarantees. Finally, we generalize our sampling scheme to accommodate *a priori* information about filters (i.e., `WHERE` clause).

In this paper, we make the following contributions:

1. We discuss two metrics—output size and estimator's variance —for measuring the quality of join approximation, and show that universe sampling is optimal for output size and there is an information-theoretical lower bound for variance (Section 5.3).
2. We formalize a hybrid scheme, called Stratified-Universe-Bernoulli Sampling (SUBS), which allows for different combinations of stratified, universe, and Bernoulli sampling. We derive optimal sampling parameters within this scheme, and show that they achieve the theoretical lower bound of variance within a constant factor (Section 5.4–5.5.3). We also extend our analysis to accommodate additional information regarding the `WHERE` clause (Section 5.6).
3. Through extensive experiments, we also empirically show that our optimal sampling parameters achieve lower error than existing sampling schemes in both centralized and decentralized scenarios (Section 5.7).

5.2 Background

In this section, we provide the necessary background on sampling-based join approximation. We also formally state our problem setting and assumptions.

5.2.1 Sampling in Databases

The following are the three main popular sampling strategies (operators) used in AQP engines and database systems.

1. **Uniform/Bernoulli Sampling.** Any strategy that samples all tuples with the same probability is considered a uniform (random) sample. Since enforcing fixed-size sampling without replacement is expensive in distributed systems, Bernoulli sampling is considered a more efficient strategy [93]. In Bernoulli sampling, each tuple is included in the sample independently, with a fixed sampling probability p. In this paper, for simplicity, we use "uniform" and "Bernoulli" interchangeably. As mentioned in Section 5.1, joining two uniform samples leads to quadratically fewer output tuples. Further, it does not guarantee an i.i.d. sample of the original join [8]: the output is a uniform sample of the join but not an independent one. Consider an arbitrary tuple of the join, say (t_1, t_2), where t_1 is from the first table and t_2 is from the second. The probability of this tuple appearing in the join of the samples is always the same value, i.e., p^2. The output is thus a uniform sample. However, the tuples are not independent: consider another tuple of the join, say (t_1, t_2') where t_2' is another tuple from the second table joining with t_1. If (t_1, t_2) appears in the output, the probability of (t_1, t_2') also appearing becomes p instead of p^2, which would be the probability if they were independent.

2. **Universe Sampling.** Given a column[2] J, a (perfect) hash function $h : J \mapsto [0, 1]$, and a sampling rate p, this strategy includes a tuple t in the table if $h(t.J) \leq p$. Universe sampling is often used for equi-joins, in which the same p value and hash function h are applied to the join columns in both tables. This ensures that when a tuple t_1 is sampled from one table, any matching tuple t_2 from the other table is also sampled, simply because $t_1.J = t_2.J \Leftrightarrow h(t_1.J) = h(t_2.J)$. This is why joining two universe samples of rate p produces p fraction of the original join output *in expectation*. The output is a uniform sample of the original join, as each join tuple appears with the same probability

[2] J can also be a set of multiple columns.

p. However, there is more dependence among the output tuples. Consider two join tuples (t_1, t_2) and (t'_1, t'_2) where t_1, t'_1, t_2, t'_2 all share the same join key. Then, if (t_1, t_2) appears, the probability of (t'_1, t'_2) also appearing will be 1. Likewise, if (t_1, t_2) does not appear, the probability of (t'_1, t'_2) appearing will be 0. Higher dependence means lower accuracy (see Section 5.3.1).

3. **Stratified Sampling.** The goal of stratified sampling is to ensure that minority groups are sufficiently represented in the sample. Groups are defined according to one or multiple columns, called the *stratified columns*. A group (a.k.a. a stratum) is a set of tuples that share the same value under those stratified columns. Given a set of stratified columns C and an integer parameter k_{tuple}, a stratified sampling is a scheme that guarantees at least k_{tuple} tuples are sampled uniformly at random from each group. When a group has fewer than k_{tuple} tuples, all of them are retained.

5.2.2 Quality Metrics

Different metrics can be used to assess the quality of a join approximation. In this paper, we focus on the following two, which are used by most AQP systems.

Output Size/Cardinality This metric is the number of tuples of the original join that also appear in the join of the samples. It is mostly relevant for exploratory usecases, where users visualize or examine a subset of the output. In other cases, where an aggregate is computed from the join output, retaining a large number of output tuples does not guarantee accurate answers (we show this in Section 5.3.1).

Variance In scenarios where an aggregate function needs to be calculated from the join output, the error of the aggregate approximation is more relevant than the number of intermediate tuples generated. For most non-extreme statistics, there are readily available unbiased estimators, e.g., Horvitz-Thompson estimator [88]. Thus, a popular indicator of accuracy is the variance of the estimator [11], which determines the size of the confidence interval given a sample size.

5.2.3 Problem Statement

In this section, we formally state the problem of sample-based join approximation. The notations used throughout the paper are listed in Table 5.1.

Notation	Definition
T_1, T_2	Two tables for the join
S_i	A sample generated from table T_i
J	Column(s) used for the join between T_1 and T_2
W	Column being aggregated (e.g., `SUM`, `AVG`)
C	Column(s) used for filters (i.e., `WHERE` clause)
$\mathcal{U}$	Set of all possible values of J
a, b	Frequency vectors of T_1 and T_2's join columns, resp.
a_v, b_v	Number of tuples with join value v in T_1 and T_2, resp.
$\hat{J}_{agg}$	Estimator for a join query with aggregate function agg
ϵ	Sampling budget w.r.t. the original table size
n_1, n_2	Number of tuples in T_1 and T_2, resp.
h	A (perfect) hash function
k_{tuple}	Minimum number of tuples to be kept per group in stratified sampling
k_{key}	Minimum number of join keys per group to apply universe sampling (universe sampling is not applied to groups with fewer than k_{key} join keys)
p	Sampling rate of universe sampling
q	Sampling rate of uniform sampling

Table 5.1: Notations.

Query Estimator Let S_1 and S_2 be two samples generated offline from tables T_1 and T_2, respectively, and q_{agg} be a query that computes an aggregate function agg on the join of T_1 and T_2. A query estimator $\hat{J}_{agg}(S_1, S_2)$ is a function that estimates the value of agg using two samples rather than the original tables.

Join Sampling Problem Given a query estimator $\hat{J}_{agg}$ and a sampling budget $\epsilon \in (0, 1]$, our goal is to create a pair of samples S_1 and S_2—from tables T_1 and T_2, respectively— that are optimal in terms of a given success metric, while respecting a given storage budget *epsilon* on average. Specifically, we seek S_1 and S_2 that minimize $\hat{J}_{agg}$'s variance or maximize its output size such that $E[|S_1| + |S_2|] \leq \epsilon \times (|T_1| + |T_2|)$.

To formally study this problem, we first need to define a class of reasonable sampling strategies. In Section 5.4, we define a hybrid scheme that can capture different combinations of stratified, universe, and uniform sampling.

5.2.4 Scope and Limitations

To simplify our analysis, we limit our scope in this paper.

Flat Equi-joins We focus on equi (inner) joins as the most common form of joins in practice. We also support both `WHERE` and `GROUPBY` clauses. Because our focus is on the join itself, we ignore nested queries and only consider flat (or flattened) queries. We primarily focus on two-way joins. However, our results extend to multi-way joins with the same join column(s).

Aggregate Functions Most AQP systems do not support extreme statistics, such as `MIN` or `MAX` [112]. Likewise, we only consider non-extreme aggregates, and primarily focus on the three basic functions, `COUNT`, `SUM`, and `AVG`. However, we expect our techniques to easily extend to other mean-like statistics as well, such as `VAR`, `STDEV`, and `PERCENTILE`.

5.3 Hardness

In this section, we explain why providing a large output size is insufficient for approximating joins, and formally show the hardness of approximating common aggregates based on the theory of communication complexity.

5.3.1 Output Size

Uniform sampling leads to small output size. If we sample at a rate q from both table T_1 and table T_2, the join of samples contains only q^2 fraction of $T_1 \bowtie T_2$ in expectation. Moreover, the join of two independent samples of the original tables is in general not an independent sample of $T_1 \bowtie T_2$, which hurts the sample quality. In contrast, universe sampling [93] with sample rate p can, in expectation, sample a p fraction of $T_1 \bowtie T_2$. We prove that this is optimal:

Theorem 5.1. *No sampling scheme with sample rate α can guarantee more than α fraction of $T_1 \bowtie T_2$ in expectation for all possible inputs.*

Proof. We can simply consider two identical tables T_1, T_2 of n tuples, each having join key $1, 2, \ldots, n$. Their join has size n. Since each tuple of T_1 joins with exactly one tuple of T_2, the size of the join of the samples must not be larger than the size of sample of T_1. Since, by assumption, the expected size of the sample of T_1 is at most αn, the expected size of the join of the samples must also be at most αn. □

However, a large number of tuples retained in the join does not imply that the original join query can be accurately approximated. As pointed out in [35], universe sampling shows poor performance in approximating queries when the frequencies of keys are concentrated on a few elements. Consider the following extreme example with tables T_1 and T_2, each comprised of n tuples with a single value 1 in their join key. In this example, universe sampling with the sampling rate p produces an estimator of variance n^4/p, while uniform sampling with rate q has a variance of n^2/q^2, which is much lower when $p = q$ and n is large. Thus, a larger output size does not necessarily lead to a better approximation of the query.

5.3.2 Approximating Aggregate Queries

In this section, we focus on the core question: why is approximating common aggregates (e.g., `COUNT`, `SUM` and `AVG`) hard when using a small sample (or more generally, a small summary)? We address this question using the theory of communication complexity. Specifically, to show that computing `COUNT` on a join is hard, we reduce it to set intersection, a canonically hard problem in communication complexity. Consider two parties, Alice and Bob, holding information x and y, respectively. They want to evaluate some function $f(x, y)$ while exchanging as little information as possible. Assume that both Alice and Bob each hold a set of size k, say A and B, respectively. They aim to estimate the size of $t = |A \cap B|$. Pagh et. al [117] show that if Alice only sends a small summary to Bob, any unbiased estimator that Bob uses will have a large variance.

Theorem 5.2 (See [117])**.** *Any one-way communication protocol that estimates t within relative error δ with probability at least $2/3$ must send at least $\Omega(k/(t\delta^2))n$ bits.*

Corollary 5.3. *Any estimator to $|A \cap B|$ produced by Bob that is based on an s-bits summary by Alice must have a variance of at least $\Omega(kt/s)$.*

Any sample of size s can be encoded using $O(\log \binom{k}{s})$ bits, implying that any estimator to `COUNT` that is based on a sample of size s from one of the tables must have a variance of at least $\Omega(kt/s)$.

Estimating `SUM` queries is at least as hard as estimating `COUNT` queries, since any `COUNT` can be reduced to a `SUM` by setting all entries in the `SUM` column to 1.

From the hard instance of set intersection, we can also derive a hard instance for `AVG` queries. Based on Theorem 5.2, any summary of T_1 that can distinguish between

intersection size $t(1+\delta)$ and $t(1-\delta)$ must be at least of size $\Omega(k/(t\delta^2))$ bits. Now we reduce this problem to estimating an `AVG` query.

Here, the two tables consist of $k+\sqrt{t}$ tuples each. The first k tuples of T_1 and T_2 are from the hard instance of set intersection, and the values of their `AVG` column are set to $2r$. The join column of the last $\sqrt{t}$ tuples is set to some common key v' that is in the first k tuples, and their `AVG` column is set to 0. Therefore, the intersection size from the first k tuples is at least $t(1+\delta)$ (or at most $(t(1-\delta))$) if and only if the result of the `AVG` query is at least $\frac{2rt(1+\delta)}{t(2+\delta)} = (1+O(\delta))r$ (or at most $\frac{2rt(1+\delta)}{t(2+\delta)} = (1-O(\delta)r)$). By re-scaling δ by a constant factor, we can get the following theorem:

Theorem 5.4. *Any summary of T_1 that can estimate an `AVG` query with precision δ with probability at least $2/3$ must have a size of at least $\Omega(n/(t\delta^2))$.*

5.4 Generic Sampling Scheme

To formally argue about the optimality of a sampling strategy, we must first define a *class* of sampling schemes. As discussed in Section 5.2.1, there are three well-known sampling operators: stratified, universe, and Bernoulli (uniform). However, these atomic operators can themselves be combined. For example, one can apply universe sampling of rate 0.1 and then Bernoulli sampling of rate 0.2 for an overall effective sampling rate of 0.02.[3] To account for such hybrid schemes, we define a generic scheme that combines universe and Bernoulli sampling, called UBS.[4] We also define a more generic scheme that combines all three of stratified, universe and Bernoulli sampling, called SUBS. It is easy to show that the basic sampling operators are a special case of SUBS. First, we define the effective sample rate.

Definition 5.5 (Effective sampling rate)**.** *We define the effective sampling rate of a sampling scheme as the expected ratio of the size of the resulting sample to that of the original table.*

Definition 5.6 (Universe-Bernoulli Sampling (UBS))**.** *Given a table T and a column (or set of columns) J in T, a UBS scheme is defined by a pair (p, q), where $0 < p \leq 1$*

[3]Statistically, it does not matter which sampling is applied first: whether a tuple passes the universe sampler and whether it passes the Bernoulli sampler are completely independent decisions, and hence, the output distribution is the same. Here, we apply universe sampling first only for convenience and without loss of generality.

[4]Even if we do not care about output cardinality, universe sampling can still help improve the approximation quality. For example, given two tables of size n with a one-to-one join relationship, the count estimator's variance is n/q^2 under Bernoulli sampling but n/p under universe sampling, which is much lower when $p=q$.

is a universe sampling rate and $0 < q \leq 1$ *is a Bernoulli (or uniform) sampling rate. Let* $h : \mathcal{U} \mapsto [0, 1]$ *be a perfect hash function. Then, a sample of* T *produced by this scheme,* $S = UBS_{p,q}(T, J)$*, is produced as follows:*

1: **function** $\text{UBS}_{p,q}((T, J))$
2: $S \leftarrow \emptyset$
3: **for** each $t \in T$ **do**
4: **if** $h(t.J) \leq p$ **then**
5: Include t into S independently with probability q.
6: **end if**
7: **end for**
8: **end function**

It is easy to see that the effective sampling rate of a UBS scheme (p, q) is $p \cdot q$. Thus, the effective sampling rate here is independent of the actual distribution of the values in the table (and column(s) J).

The goal of this sampling paradigm is to optimize the trade-off between universe sampling and Bernoulli sampling in different instances. At one extreme, when each join value appears exactly once in both table, universe sampling leads to lower variance than Bernoulli sampling. This is because independent Bernoulli sampling has trouble matching tuples with the same join value, while universe sampling guarantees that when a tuple is sampled, all matching tuples in the other table are also sampled. At the other extreme, if all tuples have the same join value in both tables (i.e., the join becomes a Cartesian product of the two tables), universe sampling will either sample the entire join, or sample nothing at all, while uniform sampling will have a sample size concentrated around qN, thus giving an estimator of much lower variance. In section 5.5.1 to 5.5.3, we give a comprehensive discussion on how to optimize p and q for different tables and different queries.

The Stratified-Universe-Bernoulli Sampling Scheme applies to a table T that is divided into K groups (i.e., strata), denoted as $G_1, G_2, \ldots, G_k$.

Definition 5.7 (Stratified-Universe-Bernoulli Sampling (SUBS))**.** *Given a table* T *of* N *rows and a column (or set of columns)* J *in* T*, a SUBS scheme is defined by a tuple* $(p_1, p_2, \ldots, p_K, q_1, q_2, \ldots, q_K)$*, where* $0 < p_i, q_i \leq 1$ *are the universe and Bernoulli sampling rates. Given a perfect hash function* h*:* $\mathcal{U} \mapsto [0, 1]$*, a sample of* T *produced by this scheme,* $S = UBS_{p,q}(T, J)$*, is produced as follows:*

```
function SUBS_{p_1,p_2,...,p_K,q_1,q_2,...,q_K}((T, G, J))
    S ← ∅
    for each group G_i do
        for each tuple t ∈ G_i do
            if h(t.J) ≤ p_i then
                Include t into S independently with probability q_i.
            end if
        end for
    end for
end function
```

Let $|G_i|$ denote the number of tuples in group G_i. Then the effective sampling rate of a SUBS scheme is $\sum_i p_i \cdot q_i \cdot |G_i|/N$. We call $\epsilon_i = p \cdot q_i$ the effective sampling rate for group G_i.

In both UBS and SUBS schemes, the user specifies ϵ as their desired sampling budget, given which our goal is to determine optimal sampling parameters p and q (or p_i and q_i values) such that the variance of our join estimator is minimized. In Section 5.5, we derive the optimal p and q for UBS. For SUBS, in addition to ϵ, the user also provides two additional parameters k_{key} and k_{tuple} (explained below). Next, we show how to determine the effective sampling rate ϵ_i for each group G_i based on these parameters in SUBS. Given ϵ_i for each group, the problem is then reduced to finding the optimal parameters for UBS for that group (i.e., p_i and q_i). Moreover, as we will show in Sections 5.5.1–5.5.3, particularly in Lemma 5.9, the universe sampling rate for every group must be the same, and must be the same as the universe sampling rate of the other table in two-way joins. Hence, we use a single universe sampling rate $p = p_1 = ... = p_k$ across all groups.

As mentioned in Section 5.2.1, k_{tuple} is a user-specified lower bound on the minimum number of tuples[5] in each group the sample must retain. k_{key} is an additional user-specified parameter required for the SUBS scheme. It specifies a threshold at which to activate the universe sampler. In particular, if a group contains too few (i.e., less than k_{key}) *join keys*, we do not perform any universe sampling as it will have a high chance of filtering out all tuples. Hence, we apply universe sampling only to those groups with $\geq k_{\text{key}}$ join keys. For groups with fewer than k_{key} join keys, we will only apply Bernoulli sampling with rate ϵ_i.

We call a group *large* if it contains at least k_{key} join keys, otherwise, we call it a

[5]The lower bound holds only on average, due to the probabilistic nature of sampling.

small group. We use N_b to denote the total number of tuples in all large groups, and N_s to denote the total number of tuples in all small groups. Similarly, let M_b and M_s denote the number of large and small groups, respectively. Then, we decide the sampling budget ϵ_i for each group G_i as follows:

1. If $M_s k_{\text{tuple}} > \epsilon N_s$ or $M_b k_{\text{tuple}} > \epsilon N_b$, we notify the user that creating a sample given their parameters is infeasible.

2. Otherwise,

   - Let $\epsilon'_s = \frac{K_s \cdot k_{\text{tuple}}}{N_s}$ and let $\epsilon''_s = \epsilon - \epsilon'_s$. Then for each small group G_i, the sampling budget is $\epsilon_i = \frac{k_{\text{tuple}}}{|G_i|} + \epsilon''_s$.
   - Let $\epsilon'_b = \frac{K_b \cdot k_{\text{tuple}}}{N_b}$ and let $\epsilon''_b = \epsilon - \epsilon'_b$. Then for each large group G_i, the sampling budget is $\epsilon_i = \frac{k_{\text{tuple}}}{|G_i|} + \epsilon''_b$.

Once ϵ_i is determined for each group, the problem of deciding optimal SUBS parameters is reduced to deciding the optimal SUBS parameters for K separate groups. This effective sampling rate ϵ_i guarantees that each large group will have at least t tuples in the sample on average, and the remaining budget is divided evenly. Thus, the corresponding uniform sampling rate for each large group is $q_i = \epsilon_i / p$. Moreover, we pose the constraint that the universe sampling rate p should be at least $1/s$ to guarantee that, on average, there is at least one join key passing through the universe sampler.

For small groups, we simply apply uniform sampling with rate ϵ_i. This is equivalent to setting $p = 1$ for these groups.

Overall, this strategy provides the following guarantees:

1. Each group will have at least t tuples in the sample, on average.

2. The probability of each group being missed is at most $(1 - 1/s)^s < 0.367$. In general, if we set $p > c/s$ for some constant $c > 1$, this probability will become 0.367^c.

3. The approximation of the original query will be optimal in terms of its variance (see Sections 5.5.1–5.5.3).

5.5 Optimal Sampling

As shown in Section 5.4, finding the optimal sampling parameters within the SUBS scheme can be reduced to finding those within the UBS scheme. Thus, in this section,

we focus on deriving the UBS parameters that minimize error for each aggregation type (`COUNT`, `SUM`, and `AVG`). Initially, we also assume there is no `WHERE` clause. Later, in Section 5.6, we show how to handle `WHERE` conditions and how to create a single sample instead of creating one per each aggregation type and `WHERE` condition.

Centralized vs. Decentralized For each aggregation type, we analyze two scenarios: centralized and decentralized. Centralized setting is when the frequencies of the join keys in both tables are known. This represents situations where both tables are stored on the same server, or each server communicates its full frequency statistics to other parties. Decentralized setting is a scenario where the two tables are each stored on a separate server [144], and exchanging full frequency statistics across the network is costly.[6]

Decentralized Protocols In a decentralized setting, each party (i.e., server) only has access to full statistics of its own table (e.g., frequencies, join column distribution). The goal then is for each party to determine its sampling strategy, while minimizing communications with the other party. Depending on the amount of information exchanged, one can pursue different protocols for achieving this goal. In this paper, we study a simple sampling protocol, which we call DICTATORSHIP. Here, one server, say `party1`, is chosen as the dictator. We also assume that the parties know each other's sampling budgets and table sizes (ϵ_1, ϵ_2, $|T_1|$, and $|T_2|$). The dictator observes the distributional information of its own table, say T_1, and decides a shared universe sampling rate p between $\max\{\epsilon_1, \epsilon_2\}$ and 1. This p is sent to the other server (`party2`) and both servers use p as their universe sampling rate.[7] Their uniform sampling rates will thus be $q_1 = \epsilon_1/p$ and $q_2 = \epsilon_2/p$, respectively.

Since `party1` only has T_1's frequency information, it chooses an optimal value of p that minimizes the *worst case* variance of $\hat{J}_{agg}$, i.e., the variance when the frequencies in T_2 are chosen adversarially. This can be formulated as a robust optimization [113]:

$$p^* = \underset{\max\{\epsilon_1,\epsilon_2\}\leq p\leq 1}{\arg\min} \max_b \mathrm{Var}[\hat{J}_{agg}] \tag{5.1}$$

where b ranges over all possible frequency vectors of T_2.

An alternative protocol is a VOTER protocol, where each party proposes (i) a universe sampling rate and (ii) a worst case variance if this rate were to be adopted

[6]Here, we focus on two servers, but the math can easily be generalized to decentralized networks of multiple servers.

[7]Using the same universe sampling rate is justified by Lemma 5.9.

by everyone. Once this information is exchanged, all parties adopt the rate with the best worst case variance. While offering a better worst case variance by design, VOTER does not guarantee that the actual variance will indeed be lower than that of DICTATORSHIP. This is because the actual variance depends on the local statistics of the other parties as well.

Moreover, for `SUM` and `AVG` queries, which unlike `COUNT` queries involve an aggregate column, one can show that VOTER is unnecessary: it is always better to simply adopt the rate proposed by the party with has the aggregate column. This is because the party without this information can only assume an arbitrary distribution of the aggregate column, leading to overestimation of the worst case variance.

Note that, if one modifies VOTER such that each party also shares their full statistics with others, the protocol then reduces to a centralized setting but with significantly more communication.

There is yet a more complex protocol that one can apply in a decentralized setting: an iterative, multi-round protocol, called EXPLORER. In this protocol, the parties each choose an arbitrary value as their initial sampling parameter, say p_1^0 and p_2^0, and produce a sample of their own table using their own parameter, say S_1^0 and S_2^0, respectively. Then, they each share their chosen sampling rate and sample with the other party. Then, each party uses its own local table and the sample received from the other party to derive a new sampling parameter, say p_1^1 and p_2^1, respectively. This process continues iteratively until the sampling parameters converge, or the amount of communication exceeds a fixed budget. EXPLORER, however, is significantly more expensive than both DICTATORSHIP and VOTER. A full analysis of the EXPLORER protocol is beyond the scope of the current paper, and we leave to future work.

In the rest of this paper, we use DICTATORSHIP in our decentralized analysis. VOTER and EXPLORER are also viable protocols in the decentralized setting and we defer their theoretical and experimental study to any future works.

5.5.1 Join Size Estimation: Count on Joins

We start by considering the following simplified query:

```
select count(*) from T1 join T2 on J
```

where T_1 and T_2 are two tables joined on column(s) J. Consider $S_1 = \text{UBS}_{(p_1,q_1)}(T_1, J)$ and $S_2 = \text{UBS}_{(p_2,q_2)}(T_2, J)$. Then, we can define an unbiased estimator for the above query, $E_{\text{count}} = |T_1 \bowtie_J T_2|$, using S_1 and S_2 as follows. Observe that given any pair of tuples $t_1 \in T_1$ and $t_2 \in T_2$, where $t_1.J = t_2.J$, the probability that (t_1, t_2) enters

$S_1 \bowtie S_2$ is $p_{\min} q_1 q_2$, where $p_{\min}$=min$\{p_1,\ p_2\}$. Hence, the following is an unbiased estimator for E_{count}.

$$\hat{J}_{\text{count}}(p_1, q_1, p_2, q_2, S_1, S_2) = \frac{1}{p_{\min} q_1 q_2} |S_1 \bowtie S_2|. \tag{5.2}$$

When the arguments $p_1, q_1, p_2, q_2, S_1, S_2$ are clear from the context, we omit them and simply write $\hat{J}_{\text{count}}$.

Lemma 5.8. *Let $S_1 = UBS_{p_1,q_1}(T_1, J)$ and $S_2 = UBS_{p_2,q_2}(T_2, J)$. The variance of $\hat{J}_{count}$ is as follows:*

$$\begin{aligned}\text{Var}(\hat{J}_{count}) = \frac{1-p}{p}\gamma_{2,2} + \frac{1-q_2}{pq_2}\gamma_{2,1} \\ + \frac{1-q_1}{pq_1}\gamma_{1,2} + \frac{(1-q_1)(1-q_2)}{pq_1q_2}\gamma_{1,1}.\end{aligned}$$

where $\gamma_{i,j} = \sum_v a_v^i b_v^j$.

Proof. Let X_v and Y_v be the random variable denoting the number of tuple in S_1 and S_2 with value v given that $h(v) \leq p_{\min}$. Therefore, X_v and Y_v are binomial random variables with parameter (a_v, q_1) and (b_v, q_2). Let Z_v be defined as the number of tuples in $S_1 \bowtie S_2$ with join value v. By construction, we have:

$$Z_v \stackrel{\text{def}}{=} \begin{cases} X_v Y_v & \text{with probability } p \\ 0 & \text{otherwise} \end{cases}$$

And

$$\hat{J} = \frac{1}{pq_1q_2} \sum_{v \in \mathcal{U}} Z_v.$$

To analyze the variance of Z_v, we use the law of total variance: Let $W_v = X_v Y_v$, we have

$$\text{Var}(Z_v) = \text{E}[\text{Var}(Z_v \mid W_v) + \text{Var}(E[Z_v \mid W_v])$$

We have

$$\begin{aligned}E[\text{Var}(Z_v \mid W_v)] &= p(1-p)\text{E}[W_v^2] \\ &= p(1-p)(\text{Var}(W_v) + \text{E}^2[W_v])\end{aligned}$$

And

$$\mathrm{Var}(E[Z_v \mid W_v]) = p^2\mathrm{Var}(W_v)$$

Combining the two terms we have

$$\mathrm{Var}(Z_v) = p\mathrm{Var}(W_v) + p(1-p)\mathrm{E}^2[W_v]$$

where

$$\begin{aligned}\mathrm{Var}(W_v) =& \mathrm{E}[X_v^2]\mathrm{E}[Y_v^2] - \mathrm{E}^2[X_v]\mathrm{E}^2[Y_v] \\ =& (a_v q_1(a_v q_1 + 1 - q_1))(b_v q_2(b_v q_2 + 1 - q_2)) \\ & - q_1^2 q_2^2 a_v^2 b_v^2 \\ =& a_v^2 b_v q_1^2 q_2(1-q_2) + a_v b_v^2 q_1(1-q_1)q_2^2 \\ & + a_v b_v q_1(1-q_1)q_2(1-q_2)\end{aligned}$$

and

$$\mathrm{E}^2[W_v] = q_1^2 q_2^2 a_v^2 b_v^2.$$

Therefore,

$$\begin{aligned}\mathrm{Var}[Z_v] = & p(a_v^2 b_v q_1^2 q_2(1-q_2) + a_v b_v^2 q_1(1-q_1)q_2^2 \\ & + a_v b_v q_1(1-q_1)q_2(1-q_2)) + p(1-p)q_1^2 q_2^2 a_v^2 b_v^2\end{aligned}$$

Using our notation for frequency vectors and its moments. The variance of the join size estimator $\hat{J}$ can be written as:

$$\begin{aligned}& \mathrm{Var}(\hat{J}_{\mathrm{count}}) \\ =& \sum_{v \in U} \frac{1}{p^2 q_1^2 q_2^2}\mathrm{Var}(Z_v) \\ =& \frac{1-p}{p}\sum_v a_v^2 b_v^2 + \frac{1-q_2}{pq_2}\sum_v a_v^2 b_v \\ & + \frac{1-q_1}{pq_1}\sum_v a_v b_v^2 + \frac{(1-q_1)(1-q_2)}{pq_1 q_2}\sum_v a_v b_v)\end{aligned}$$

□

To minimize $\text{Var}(\hat{J}_{\text{count}})$ under a fixed sampling budget, the two tables should always use the same *universe* sampling rate. If $p_1 > p_2$, the *effective universe sampling rate* is only p_2, i.e., only p_2 fraction of the join keys inside T_1 appear in the join of the samples, and the remaining $p_1 - p_2$ fraction is simply *wasted*. Then, we can change the universe sampling rate of T_1 to p_2 and increase its uniform sampling rate to obtain a lower variance.

Lemma 5.9. *Given tables T_1, T_2 joined on column(s) J, a fixed sampling parameter (p_1, q_1) for T_1, and a fixed effective sampling rate ϵ_2 for T_2, the variance of $\hat{J}_{count}$ is minimized when T_2 uses p_1 as its universe sampling rate and correspondingly ϵ_2/p_1 as its uniform sampling rate.*

Proof. Define p_2 and q_2 to be the universe and Bernoulli sampling rate for T_2 and $p = \min\{p_1, p_2\}$. For fixed p_1, q_1 and ϵ_2, we write $\text{Var}[\hat{J}_{\text{count}}]$ as a function of p_2.

If $p_2 \geq p_1$, we have $p = p_1$:

$$
\begin{aligned}
&\text{Var}(\hat{J}_{\text{count}}) \\
=&\frac{1-p}{p}\sum_v a_v^2 b_v^2 + \frac{1-q_2}{pq_2}\sum_v a_v^2 b_v \\
&+ \frac{1-q_1}{pq_1}\sum_v a_v b_v^2 + \frac{(1-q_1)(1-q_2)}{pq_1q_2}\sum_v a_v b_v \\
=&\frac{1-p}{p}\sum_v a_v^2 b_v^2 + (1\epsilon_2 - \frac{1}{p_2})\sum_v a_v^2 b_v \\
&+ \frac{1-q_1}{q_1}\sum_v a_v b_v^2 + \frac{1-q_1}{q_1}(\frac{p_2}{\epsilon} - 1)\sum_v a_v b_v)
\end{aligned}
$$

which is nondecreasing in terms of p_2. Therefore, the minimum is attained when $p_2 = p_1$. Intuitively, increasing p_2 when $p_2 \geq p_1$ decreases the Bernoulli sampling rate without increasing the universe sampling rate over join, and Hence only decrease the quality of samples.

When $p_2 \leq p_1$, $p = p_2$:

$$\begin{aligned}
&\mathrm{Var}(\hat{J}_{\mathrm{count}}) \\
=&\frac{1-p}{p}\sum_v a_v^2 b_v^2 + \frac{1-q_2}{pq_2}\sum_v a_v^2 b_v \\
&+ \frac{1-q_1}{pq_1}\sum_v a_v b_v^2 + \frac{(1-q_1)(1-q_2)}{pq_1q_2}\sum_v a_v b_v \\
=&(\frac{1}{p_2}-1)\sum_v a_v^2 b_v^2 + (\frac{1}{\epsilon_2}-\frac{1}{p_2})\sum_v a_v^2 b_v \\
&+ \frac{1}{p_2}\frac{1-q_1}{q_1}\sum_v a_v b_v^2 + \frac{1}{\epsilon_2}\frac{1-q_1}{q_1}(1-\frac{\epsilon_2}{p_2})\sum_v a_v b_v \\
=&\frac{1}{p_2}\sum_v (a_v^2 b_v^2 - a_v^2 b_v) + \frac{1}{p_2}\frac{1-q_1}{q_1}\sum_v (a_v b_v^2 - a_v b_v) \\
&- \sum_v a_v^2 b_v^2 + \frac{1}{\epsilon_2}\sum_v a_v^2 b_v - \frac{1}{\epsilon_2}\frac{1-q_1}{q_1}\sum_v a_v b_v
\end{aligned}$$

Since $0 < q_1 \geq 1$, and $a_v^2 b_v^2 - a_v^2 b_v, a_v b_v^2 - a_v b_v \geq 0$ since both a_v and b_v are nonnegative integers. The variance function is nonincreasing in terms of p_2, thereby attains its minimum when $p_2 = p_1$. □

Note that Lemma 5.9 applies to both centralized and decentralized settings, i.e., it applies to any feasible sampling parameter (p_1, q_1) and (p_2, q_2), regardless of how the sampling parameter is decided. Next, we analyze each setting.

5.5.1.1 Centralized Sampling for Count

Since in optimal sampling scheme, we always use the same (say p) universe sampling scheme across the tables, we have the following result.

Theorem 5.10. *When T_1 and T_2 use sampling parameters $(p, \epsilon_1/p)$ and $(p, \epsilon_2/p)$, $\hat{J}_{count}$'s variance is given by:*

$$\begin{aligned}
\mathrm{Var}[\hat{J}_{count}] = &(\frac{1}{p}-1)\gamma_{2,2} + (\frac{1}{\epsilon_2}-\frac{1}{p})\gamma_{2,1} \\
&+ (\frac{1}{\epsilon_1}-\frac{1}{p})\gamma_{1,2} + (\frac{p}{\epsilon_1\epsilon_2}-\frac{1}{\epsilon_1}-\frac{1}{\epsilon_2}+\frac{1}{p})\gamma_{1,1}.
\end{aligned}$$

Proof. This is simply obtained by plugging in $q_1 = \epsilon_1/p$ and $q_2 = \epsilon_2/p$ to Lemma 5.8. □

Since each term in Theorem 5.10 that depends on p is proportional either to p or $1/p$, to find a p that minimizes the variance, one can simply set the first order derivatives (with respect to p) to 0.

Theorem 5.11. *Let T_1 and T_2 be two tables joined on column(s) J. Let a_v and b_v be the frequency of value v in column(s) J of tables T_1 and T_2, respectively. Given their sampling rates ϵ_1 and ϵ_2, the optimal sampling parameters (p_1, q_1) and (p_2, q_2) are given by:*

$$p_1{=}p_2{=}\min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\frac{\epsilon_1\epsilon_2\gamma_{2,2} - \gamma_{1,2} - \gamma_{2,1} + \gamma_{1,1}}{\gamma_{1,1}}}\}\}$$

and $q_1{=}\epsilon_1/p$, $q_2{=}\epsilon_2/p$.

Proof. By Lemma 5.9, the two table has equal universe sampling rate in the optimal sampling scheme. Thus we assume $p_1 = p_2 = p$ and p is a real number between $\max\{\epsilon_1, \epsilon_2\}$ and 1, and $q_1 = \epsilon_1/p$ and $p_2 = \epsilon_2/p$. Thus we have:

$$\begin{aligned}
&\mathrm{Var}(\hat{J}_{\mathrm{count}}) \\
=&\frac{1-p}{p}\sum_v a_v^2 b_v^2 + (\frac{1}{\epsilon_2} - \frac{1}{p})\sum_v a_v^2 b_v + (\frac{1}{\epsilon_1} - \frac{1}{p})\sum_v a_v b_v^2 \\
&+ (\frac{p}{\epsilon_1\epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} - \frac{1}{p})\sum_v a_v b_v \\
=&\frac{1}{p}(\sum_v a_v^2 b_v^2 - \sum_v a_v^2 b_v - \sum_v a_v b_v^2 + \sum_v a_v b_v) \\
&+ \frac{p}{\epsilon_1\epsilon_2}\sum_v a_v b_v - \sum_v a_v^2 b_v^2 + \frac{1}{\epsilon_2}\sum_v a_v^2 b_v \\
&+ \frac{1}{\epsilon_1}\sum_v a_v b_v^2 - (\frac{1}{\epsilon_1} - \frac{1}{\epsilon_2})\sum_v a_v b_v
\end{aligned}$$

Notice only the first two terms $\frac{1}{p}(\sum_v a_v^2 b_v^2 - \sum_v a_v^2 b_v - \sum_v a_v b_v^2 + \sum_v a_v b_v) + \frac{1}{q}\frac{1}{\epsilon_1\epsilon_2}\sum_v a_v b_v$ depends on q. Moreover, both terms has nonnegative coefficients:

$$\begin{aligned}
&\sum_v a_v^2 b_v^2 - \sum_v a_v^2 b_v - \sum_v a_v b_v^2 + \sum_v a_v b_v \\
=&\sum_v (a_v^2 - a_v)(b_v^2 - b_v) \\
\geq&0 \qquad\qquad (a_v, b_v \text{ are nonnegative integers.})
\end{aligned}$$

Since p takes on value between $\max\{\epsilon_1, \epsilon\}$ and 1, by AM-GM inequality and

monotonicity of the variance function, the term is minimized when

$$p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\frac{\epsilon_1\epsilon_2 \sum_v (a_v^2 b_v^2 - a_v^2 b_v - a_v b_v^2 + a_v b_v)}{\sum_v a_v b_v}}\}\}$$

□

Substituting this into Lemma 5.8, the resulting variance is only a constant factor of Theorem 5.2's theoretical limit. For instance, consider a primary-key-foreign-key join query where $a_v \in \{0, 1\}$ and b_v is smaller than some constant, say 5, and $\epsilon_1{=}\epsilon_2{=}\epsilon$ for any ϵ, Theorem 5.11 chooses $p_1{=}p_2{=}\epsilon$. Then the variance given by Theorem 5.10 becomes $(1/\epsilon - 1)J$ where $J = \sum_v a_v b_v$ is the size of the join. Since ϵ is the expected ratio of the sample to table size, the expression $(1/\epsilon - 1)J$ matches the lower bound in Corollary 5.3 except for a constant factor.

5.5.1.2 Decentralized Sampling for Count

Motivated by Lemma 5.9, the DICTATORSHIP protocol uses the same universe sampling rate p for both parties in the decentralized setting, by solving the following robust optimization problem:

$$\arg\min_{\max\{\epsilon_1,\epsilon_2\} \le p \le 1} \max_b \mathrm{Var}[\hat{J}_{\text{count}}]$$

Based on Lemma 5.8 and 5.11, given the effective sampling rates ϵ_1 and ϵ_2, we can express $\mathrm{Var}[\hat{J}_{\text{count}}]$ as a function of frequencies $\{a_v\}$ and $\{b_v\}$, and universe sampling rate p as follows.

$$\begin{aligned}\mathrm{Var}[\hat{J}_{\text{count}}] = (\frac{1}{p} - 1)\gamma_{2,2} + (\frac{1}{\epsilon_2} - \frac{1}{p})\gamma_{2,1} \\ +(\frac{1}{\epsilon_1} - \frac{1}{p})\gamma_{1,2} + (\frac{p}{\epsilon_1\epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} + \frac{1}{p})\gamma_{1,1}.\end{aligned} \tag{5.3}$$

Lemma 5.12. *Let a_* be the maximum frequency in table T_1, v_* be any value that has that frequency, and n_b be the total number of tuples in T_2. The optimal value for $\max_{\boldsymbol{b} \in \mathcal{K}_{n_b}} \mathrm{Var}[\hat{J}_{count}]$ is given by $(\frac{1}{p} - 1)a_*^2 n_b^2 + (\frac{1}{\epsilon_2} - \frac{1}{p})a_*^2 n_b + (\frac{1}{\epsilon_1} - \frac{1}{p})a_* n_b^2 + (\frac{p}{\epsilon_1\epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} - \frac{1}{p})a_* n_b$*

Proof. Since $\mathrm{Var}[\hat{J}_{\text{count}}]$ is strictly convex as a function b_v's in its domain. To maximize this function, it suffices to consider only the extreme points in its feasible

polytope. These are the all 0 vector $\mathbf{0}$, and the vector $\hat{\boldsymbol{b}}_v$ for each join key $v \in \mathcal{U}$ that has n_b at its v-th entries and 0 everywhere else. It is easy to see that since all coefficients for are positive, the maximizing value is achieved by the vector $\hat{\boldsymbol{b}}_{v^*}$ □

In equation (5.3), given $\{a_v\}$ and a fixed p, the variance is a convex function of the frequency vector $\{b_v\}$. Thus, the frequency vector $\{b_v\}$ that maximizes the variance, i.e., the worst case $\{b_v\}$, is one where exactly one join key has a non zero frequency. This join key should be the one with the maximum frequency in T_1. This is not a representative case and using it to decide a sampling rate might drastically hinder the performance on average. We therefore require that both servers also share a simple piece of information regarding the *maximum frequency* of the join keys in each table, say $F_a = \max_v a_v$ and $F_b = \max_v b_v$. With this information, the new optimal sampling rate is given by:

Theorem 5.13. *Given ϵ_1 and ϵ_2, the optimal UBS parameter (p, q_1) and (p, q_2) for* `COUNT` *in the decentralized setting are given by*

$$p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\epsilon_1\epsilon_2(F_aF_b - F_a - F_b + 1)}\}\}$$

and $q_1 = \epsilon_1/p$, $q_2 = \epsilon_2/p$.

Proof. Consider the variance of our count estimator given by Theorem 5.10:

$$\begin{aligned}\mathrm{Var}[\hat{J}_{\mathrm{count}}] = (\frac{1}{p} - 1)\sum_v a_v^2 b_v^2 + (\frac{1}{\epsilon_2} - \frac{1}{p})\sum_v a_v^2 b_v \\ +(\frac{1}{\epsilon_1} - \frac{1}{p})\sum_v a_v b_v^2 + (\frac{p}{\epsilon_1\epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} + \frac{1}{p})\sum_v a_v b_v.\end{aligned} \tag{5.4}$$

Fixed any p, under the constraint that for all v, $a_v \le F_a$ and $b_v \le F_b$, the variance is maximized when $a_v = F_a$ and $b_v = F_b$. This defines the worst case input under such constraint. So to obtain the optimate sampling rate for the worst case input, it suffice to substitute $a_v = F_a$ and $b_v = F_b$ to Theorem 5.11, and the theorem follows. □

5.5.2 Sum on Joins

Let E_{sum} be the output of the following simplified query:

```
select sum(T1.W)
from T1 join T2 on J
```

Let F be the sum of column W in the joined samples $S_1 \bowtie S_2$. Then, the following is an unbiased estimator for E_{sum}:

$$\hat{J}_{\text{sum}} = \frac{1}{p_{\min} q_1 q_2} F \tag{5.5}$$

where $p_{\min} = \min\{p_1, p_2\}$.

We first show that the estimator is unbiased.

Lemma 5.14. $\mathrm{E}[\hat{J}_{sum}] = E_{\text{sum}}$.

Proof. Similar to $\hat{J}_{\text{count}}$, each pair of tuples (t_1, t_2) in the join appears in the join of the sample with probability $p_{\min} q_1 q_2$. We have:

$$\begin{aligned} E[\hat{J}_{\text{sum}}] &= \frac{1}{p_{\min} q_1 q_2} E[SUM_W] \\ &= \frac{1}{p_{\min} q_1 q_2} \sum_{\substack{(t_1,t_2): \\ t_1 \in T_1, t_2 \in T_2 \\ t_1.J = t_2.J}} ((p_{\min} q_1 q_2) t_1.c + (1 - p_{\min} q_1 q_2) \cdot 0) \\ &= \sum_{\substack{(t_1,t_2): t_1 \in T_1, t_2 \in T_2 \\ t_1.J = t_2.J}} t_1.c \end{aligned}$$

□

Let μ_v and σ_v^2 be respectively the mean and variance of attribute W of the tuples in S_1 that have the join value v. Further, recall that a_v is the number of tuples in T_1 with join value v. The following lemma gives the variance of $\hat{J}_{\text{sum}}$.

Lemma 5.15. *The variance of $\hat{J}_{sum}$ is given by:*

$$\begin{aligned} \mathrm{Var}[\hat{J}_{sum}] = \frac{1 - q_2}{p q_2} \beta_1 + \frac{1 - q_1}{p q_1} \beta_2 \\ + \frac{(1 - q_1)(1 - q_2)}{p q_1 q_2} \beta_3 + \frac{1 - p}{p} \beta_4 \end{aligned} \tag{5.6}$$

where $\beta_1 = \sum_v a_v^2 \mu_v^2 b_v$, $\beta_2 = a_v(\mu_v^2 + \sigma_v^2) b_v^2$, $\beta_3 = a_v(\mu_v^2 + \sigma_v^2) b_v$ and $\beta_4 = a_v^2 \mu_v^2 b_v^2$.

Proof. We analyze the variance of $\hat{J}_{\text{sum}}$ using a similar process as `COUNT(*)`. For each join value v, define X_v to be the sum of the c values in S_1, and Y_v to be the number of tuples in S_2 whose value for column J is v. Define $W_v = X_v Y_v$ and

$$Z_v = \begin{cases} W_v & \text{with probability } p \\ 0 & \text{otherwise} \end{cases}$$

We have:

$$E[X_v] = q_1 a_v \mu_v$$

and

$$E[X_v^2] = (q_1(1-q_1)a_v(\mu_v^2 + \sigma_v^2) + q_1^2 a_v^2 \mu_v^2$$

Recall that $E[Y_v] = q_2 a_v$ and $E[Y_v^2] = b_v q_2(b_v q_2 + 1 - q_2)$. Hence

$$\begin{aligned}
&\mathrm{Var}[W_v] \\
=&E[X_v^2]E[Y_v^2] - E^2[X_v]E^2[Y_v] \\
=&((q_1(1-q_1)a_v(\mu_v^2 + \sigma_v^2) \\
&+ q_1^2 a_v^2 \mu_v^2)(b_v q_2(b_v q_2 + 1 - q_2)) - q_1^2 a_v^2 \mu_v^2 q_2^2 b_v^2
\end{aligned}$$

And

$$\begin{aligned}
&\mathrm{Var}(Z_v) \\
=&p\mathrm{Var}(W_v) + p(1-p)E^2[W_v] \\
=&p((q_1(1-q_1)a_v(\mu_v^2 + \sigma_v^2) - q_1^2 a_v^2 \mu_v^2)(b_v q_2(b_v q_2 + 1 - q_2)) \\
&- q_1^2 a_v^2 \mu_v^2 q_2^2 b_v^2) + p(1-p) q_1^2 q_2^2 a_v^2 \mu_v^2 b_v^2
\end{aligned}$$

Hence

$$\begin{aligned}
&\mathrm{Var}(\hat{J}_{\mathrm{sum}}) \\
=&\frac{1}{p^2 q_1^2 q_2^2} \sum_v \mathrm{Var}(Z_v) \\
=&\frac{1-q_2}{p q_2} a_v^2 \mu_v^2 b_v + \frac{1-q_1}{p q_1} a_v(\mu_v^2 + \sigma_v^2) b_v^2 \\
&+ \frac{(1-q_1)(1-q_2)}{p q_1 q_2} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1-p}{p} a_v^2 \mu_v^2 b_v^2
\end{aligned}$$

□

Analogous to Lemma 5.9, we have the following result.

Lemma 5.16. *Given tables T_1, T_2 joined on column(s) J, fixed sampling parameters (p_1, q_1) for T_1, and a fixed effective sampling rate $\epsilon_2 \leq p_1$ for T_2, the variance of $\hat{J}_{sum}$ is minimized when T_2 also uses p_1 as its universe sampling rate and correspondingly, ϵ_2/p_1 as its uniform sampling rate.*

Proof. Similar to Lemma 5.9, we show that for any fixed p_1 between 1 and $\max_{\epsilon_1,\epsilon_2}$

and $q_1 = \epsilon_1/p_1$, the variance of the optimal sampling parameters is minimized when the universe sampling p_2 rate of T_2 is the same as p_1:

Case *1*:: If $p_2 \geq p_1$: This case has simple intuition. When $p_2 \geq p_1$, the join universe sampling rate for both table $p = \min\{p_1, p_2\} = p_1$. Hence increasing p_2 beyond p_1 do not increase the join universe sampling rate, and only decreases the Bernoulli sampling rate for table T_2 and increases the variance of the overall estimator. In particular, we have $p = p_1$ and $pq_1 = \epsilon_1$, we have:

$$
\begin{aligned}
&\mathrm{Var}(\hat{J}_{\mathrm{sum}}) \\
=&\sum_v (\frac{1-q_2}{pq_2} a_v^2 \mu_v^2 b_v + \frac{1-q_1}{pq_1} a_v(\mu_v^2 + \sigma_v^2) b_v^2 \\
&+ \frac{(1-q_1)(1-q_2)}{pq_1q_2} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1-p}{p} a_v^2 \mu_v^2 b_v^2) \\
=&\sum_v (\frac{p_2}{\epsilon_2} - 1) \cdot \frac{1}{p} a_v^2 \mu_v^2 b_v + \frac{1-q_1}{pq_1} a_v(\mu_v^2 + \sigma_v^2) b_v^2 \\
&+ (\frac{p_2}{\epsilon_2} - 1) \frac{(1-q_1)}{pq_1} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1-p}{p} a_v^2 \mu_v^2 b_v^2)
\end{aligned}
$$

which is a non-decreasing function in terms of p_2 and it is minimized when $p_2 = p_1$.

Case *2*:: If $p_2 \leq p_1$: Here, a smaller p_2 can result in smaller join universe sampling rate in exchange of large Bernoulli sampling rate for T_2, We want to show overall decreasing p_2 will result in a large variance of $\hat{J}_{\mathrm{sum}}$. In this case, $p = p_2$ and $pq_2 = \epsilon$,

we have:

$$
\begin{aligned}
&\mathrm{Var}(\hat{J}_{\mathrm{sum}}) \\
=&\sum_v (\frac{1}{p} a_v^2 \mu_v^2 b_v^2 + \frac{1-q_2}{pq_2} a_v^2 \mu_v^2 b_v + \frac{1-q_1}{pq_1} a_v (\mu_v^2 + \sigma_v^2) b_v^2 \\
&+ \frac{(1-q_1)(1-q_2)}{pq_1q_2} a_v (\mu_v^2 + \sigma_v^2) b_v - a_v^2 \mu_v^2 b_v^2) \\
=&\sum_v (\frac{1}{p_2} a_v^2 \mu_v^2 b_v^2 + (\frac{1}{\epsilon_2} - \frac{1}{p_2}) a_v^2 \mu_v^2 b_v \\
&+ \frac{1}{p_2} \cdot \frac{1-q_1}{q_1} a_v (\mu_v^2 + \sigma_v^2) b_v^2 \\
&+ (\frac{1}{\epsilon_2} - \frac{1}{p_2}) \frac{(1-q_1)}{q_1} a_v (\mu_v^2 + \sigma_v^2) b_v - a_v^2 \mu_v^2 b_v^2) \\
=&\sum_v (\frac{1}{p_2} a_v^2 \mu_v^2 (b_v^2 - b_v) + \frac{1}{\epsilon_2} a_v^2 \mu_v^2 b_v \\
&+ \frac{1}{p_2} \frac{1-q_1}{q_1} a_v (\mu_v^2 + \sigma_v^2) (b_v^2 - b_v) \\
&+ \frac{1}{\epsilon_2} \frac{(1-q_1)}{q_1} a_v (\mu_v^2 + \sigma_v^2) b_v - a_v^2 \mu_v^2 b_v^2)
\end{aligned}
$$

Since b_v's are nonnegative integers so $b_v^2 \geq b_v$, $\mathrm{Var}(\hat{J}_{\mathrm{sum}})$ is a non-increasing function in terms of p_2 and is minimized when $p_2 = p_1$. □

5.5.2.1 Centralized Sampling for Sum

Based on Lemma 5.16, we use the same universe sampling rate $p \geq \epsilon_1, \epsilon_2$ for both tables, with their corresponding uniform sampling rates being $q_1 = \epsilon_1/p$ and $q_2 = \epsilon_2/p$. Then we can further simplify equation 5.6 into:

Theorem 5.17. *When T_1 and T_2 both use the universe sampling rate p and respectively use the uniform sampling rate $q_1 = \epsilon_1/p$ and $q_2 = \epsilon_2/p$, the variance of $\hat{J}_{sum}$ is given by:*

$$
\begin{aligned}
\mathrm{Var}[\hat{J}_{sum}] =& \sum_v (\frac{1}{\epsilon_2} - \frac{1}{p}) \beta_1 + (\frac{1}{\epsilon_1} - \frac{1}{p}) \beta_2 \\
&+ (\frac{p}{\epsilon_1 \epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} + \frac{1}{p}) \beta_3 + (\frac{1}{p} - 1) \beta_4.
\end{aligned}
$$

Proof. Similar to Theorem 5.10, this expression can be obtained by substituting $q_1 = \epsilon_1/p$ and $q_2 = \epsilon_2/p$ back to the variance given in Theorem 5.15. □

The proof of the following theorem is similar to Theorem 5.11.

Theorem 5.18. *Given effective sampling rates ϵ_1, ϵ_2, the optimal sampling parameters for* `SUM` *in a centralized setting are given by* $p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\epsilon_1\epsilon_2 \frac{\beta_1+\beta_3-\beta_2-\beta_4}{\beta_3}}\}\}$, $q_1 = \frac{\epsilon_1}{p}$ *and* $q_2 = \frac{\epsilon_2}{p}$.

Proof. This is analogous to Theorem 5.11. Since p takes on value between $\max\{\epsilon_1, \epsilon\}$ and 1, by AM-GM inequality and monotonicity of the variance function, the variance is minimized when

$$\begin{aligned} p = \min\{1, \max\{\epsilon_1, \epsilon_2, \\ (\epsilon_1\epsilon_2(\sum_v (a_v^2\mu_v^2 b_v^2 - a_v^2\mu_v^2 b_v - a_v(\mu_v^2 + \sigma_v^2)b_v^2 \\ + a_v(\mu_v^2 + \sigma_v^2)b_v))/(\sum_v a_v(\mu_v^2 + \sigma_v^2)b_v))^{1/2}\}\}. \end{aligned}$$

□

5.5.2.2 Decentralized Sampling for Sum

Lemma 5.16 implies that, in a decentralized setting for `SUM` estimation, the universe sampling rate p must be decided by the party that has T_1, i.e., the table with the aggregate column.

Given a fixed T_1 and p, $\mathrm{Var}[\hat{J}_{\mathrm{sum}}]$ is a strictly convex function of T_2's frequency vector. Hence, the worst case instance is a point distribution where all tuples in T_2 share the same join key. However, for `SUM`, the worst case distributions in T_2 are *not* the same for all possible sampling parameters p. Define $h_v(p)$ to be $\mathrm{Var}[\hat{J}_{\mathrm{sum}}]$ as a function of p where T_2's frequency vector is all concentrated on the join key v, and define $h^*(p) = \max_v h_v(p)$. Since all $h_v(p)$'s are convex in p, $h^*(p)$ is still convex and its exact minimum can be computed using a sweep line algorithm (see [42, §8] for details). In a nutshell, the algorithm sweeps all possible values of p and uses a data structure to keep track of $\max_v h_v(p)$ at that particular value. The data structure uses $O(|\mathcal{U}|)$ memory, which can be costly in practice.

Therefore, we propose a simple sampling scheme whose worst case variance is at most twice the variance of the optimal scheme. Instead of using $h^*(p)$ to keep track of the maximum of all $h_v(p)$, we use an approximate $h'(p) = \max\{h_{v_1}(p), h_{v_2}(p)\}$, where $v_1 = \arg\max_v a_v^2\mu^2$ and $v_2 = \arg\max_v a_v(\mu_v^2 + \sigma_v^2)$ to approximate $h^*(p)$. Since $h_v(p)$ is a function in the form of $h(p) = Ap + B/p + C$ for some constant $A, B, C > 0$, the value of $p^* = \arg\min_{\{\epsilon_1, \epsilon_2 \le p \le 1\}} h'(p)$ can be easily solved using quadratic equations

and basic case analysis. The function h' is much simpler and its minimum can be easily found using quadratic equations and basic case analysis.

Let $p' = \arg\min h'(p)$ and $p^* = \arg\min h^*(p)$. We claim that choosing p' as our sampling parameter can only increase the optimal worst case variance by a factor of 2. This is follows from the simple fact that $h_{v_1}(p)$ uppers bounds the terms in $h_v(p)$ that depends on $a_v^2\mu_v^2$, and $h_{v_2}(p)$ upper bounds the terms that depends on $a_v(\mu_v^2 + \sigma_v^2)$. Hence their maximum is at least half of $h_v(p)$, for any v and p.

Lemma 5.19. *For any $p \geq \epsilon_1, \epsilon_2$, we have $\frac{h^*(p)}{2} \leq h'(p) \leq h^*(p)$.*

Proof. We focus on the first inequality $1/2h^*(p) \leq h'(p)$. The second inequality holds since it is $h^*(p)$ is obtained by maximizing over a larger subset.

Observe that we can group the terms in 5.6 into $f_1(\boldsymbol{b})$ and $f_2(\boldsymbol{b})$, where

$$f_1(\boldsymbol{b}, p) = \sum_v (\frac{1}{\epsilon_2} - \frac{1}{p}) a_v^2 \mu_v^2 b_v + (\frac{1}{p} - 1) a_v^2 \mu_v^2 b_v^2$$

and

$$\begin{aligned} f_2(\boldsymbol{b}, p) =& (\frac{1}{\epsilon_1} - \frac{1}{p}) a_v (\mu_v^2 + \sigma_v^2) b_v^2 \\ &+ (\frac{p}{\epsilon_1 \epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} + \frac{1}{p}) a_v (\mu_v^2 + \sigma_v^2) b_v \end{aligned}$$

That is, f_1 consists of combinations of $a_v^2\mu_v^2$'s and f_2 consists of combinations of $a_v(\mu_v^2 + \sigma_v^2)$. Define $\boldsymbol{b}^v$ to be the vector that has n_b on its v-th coordinate and 0 everywhere else. We can rewrite h_i as:

$$h_v(p) = f_1(\boldsymbol{b}^v, p) + f_2(\boldsymbol{b}^v, p)$$

By the choice of v_1 and v_2, we have for every v that $f_1(\boldsymbol{b}^v, p) \leq f_1(\boldsymbol{b}^{v_1}, p)$ and $f_2(\boldsymbol{b}^v, p) \leq f_2(\boldsymbol{b}^{v_2}, p)$. Therefore, we have for all v that

$$f_1(\boldsymbol{b}^v, p) + f_2(\boldsymbol{b}^v, p) \leq 2 \max\{f_1(\boldsymbol{b}^{v_1}, p), f_2(\boldsymbol{b}^{v_2}, p)\}$$

Hence we have

$$\begin{aligned} h^*(p) &= \max_v f_1(\boldsymbol{b}^v, p) + f_2(\boldsymbol{b}^v, p) \\ &\leq 2 \max\{f_1(\boldsymbol{b}^{v_1}, p), f_2(\boldsymbol{b}^{v_2}, p)\} \leq 2h'(p) \end{aligned}$$

□

Corollary 5.20. *We have:* $h^*(p') \leq 2h^*(p^*)$.

5.5.3 Average on Joins

Let E_{avg} be the output of the following simplified query:

```
select avg(T1.W)
from T1 join T2 on J
```

In general, producing an unbiased estimator for `AVG` is hard.[8] Instead, we define and analyze the following estimator. Let S and C be the `SUM` and `COUNT` of column W in $S_1 \bowtie S_2$. We define our estimator as $\hat{J}_{\text{avg}} = S/C$. There are two advantages over using separate samples to evaluate `SUM` and `COUNT`: (1) we can use a larger sample to estimate both queries, and (2) since `SUM` and `COUNT` will be positively correlated, the variance of their ratio will be lower. Due to the lack of a close form expression for the variance of the ratio of two random variables, next we present a first order bivariate Taylor expansion to approximate the ratio.

For any $f(X, Y)$, the bivariate Taylor expansion around (θ_x, θ_y) is:

$$f(X,Y) = f(\theta_x, \theta_y) + f'_x(\theta_x, \theta_y)(x - \theta_x) + f'(\theta_x, \theta_y)(y - \theta_y) + R.$$

where R is a remainder of lower order terms. The expectation of $f(X, Y)$ can be approximated using the expansion around the expectation of X and Y, μ_X and μ_Y:

$$\begin{aligned}
&E[f(,Y)] \\
\approx &E[f(\mu_X, \mu_Y) + f'_x(\mu_X, \mu_Y)(x - \mu_x) + f'(\mu_x, \mu_y)(y - \mu_y)] \\
= &E[f(\mu_X, \mu_Y)] + 0 + 0 \\
= &f(\mu_X, \mu_Y)].
\end{aligned}$$

Let S and C be the random variables denoting the sum and the size of the join of samples, and let $f(X, Y) = X/Y$. This shows that $\mathrm{E}[S/C] \approx \mathrm{E}[S]/\mathrm{E}[C]$, which is exactly equal to the average over join. Although, the estimator is not unbiased, its expectation tends to the truth value when the join size tends to infinity.

Now we can analyze its variance, and with some more involved analysis we can

[8]The denominator, i.e., the size of the sampled join, can even be zero. Furthermore, the expectation of a random variable's reciprocal is not equal to the reciprocal of its expectation.

show that:

$$\text{Var}[S/C] \approx (\frac{E[S]^2}{E[C]^2})(\frac{\text{Var}[S]}{E[S]^2} - \frac{2\text{Cov}[S,C]}{E[S]E[C]} + \frac{\text{Var}[C]}{\text{E}[C]^2}). \tag{5.7}$$

Theorem 5.21. *Let S and C be random variables denoting the sum and cardinality of the join of two samples produced by applying UBS sampling parameters (p_1, q_1) to T_1 and (p_2, q_2) to T_2. Let $p_{\min} = \min\{p_1, p_2\}$. We have:*

$$\text{Var}[S/C] \approx (\frac{E[S]^2}{E[C]^2})(\frac{\text{Var}[S]}{E[S]^2} - \frac{2\text{Cov}[S,C]}{E[S]E[C]} + \frac{\text{Var}[C]}{\text{E}[C]^2}) \tag{5.8}$$

where

$$\begin{aligned}
\text{E}[S] =& p_{\min} q_1 q_2 \sum_v \mu_v a_v b_v \\
\text{E}[C] =& p_{\min} q_1 q_2 \sum_v a_v b_v \\
\text{Var}[S] =& p_{\min} q_1 q_2 (1-q_2)[q_1 \sum_v a_v^2 \mu_v^2 b_v + q_2 \sum_v a_v(\mu_v^2 + \sigma_v^2) b_v^2 \\
&+ (1-q_1) \sum_v a_v(\mu_v^2+\sigma_v^2) b_v] + p_{\min}(1-p_{\min}) q_1^2 q_2^2 a_v^2 \mu_v^2 b_v^2 \\
\text{Var}[C] =& p_{\min} q_1 q_2 [(1-q_2) \sum_v a_v^2 b_v + (1-q_1) q_2 \sum_v a_v b_v^2 \\
&+ (1-q_1)(1-q_2) \sum_v a_v b_v + (1-p_{\min}) q_1 q_2 \sum_v a_v^2 b_v^2] \\
\text{Cov}[S,C] =& p_{\min} q_1 q_2 [(1-q_2) q_1 \sum_v a_v^2 \mu_v b_v + (1-q_1) q_2 \sum_v a_v \mu_v b_v^2 \\
&+ (1-q_1)(1-q_2) \sum_v a_v \mu_v b_v + (1-p_{\min}) q_1 q_2 \sum_v a_v^2 \mu_v b_v^2]
\end{aligned}$$

Proof. For any function $f(x, y)$, the bivariate first order Taylor expansion about any (x_0, y_0) is

$$f(x,y) = f(x_0,y_0) + \frac{\partial f}{\partial x}(x_0,y_0)(x-x_0) + \frac{\partial f}{\partial y}(x_0,y_0)(y-y_0) + R$$

where R is a remainder of smaller order terms. Consider X, Y as two random variables with mean μ_x and μ_y, , we can approximate $E[f(X,Y)]$ by a expanding $E[f(x,y)]$

around (μ_x, μ_y):

$$\begin{aligned} E[f(X,Y)] &\approx E[f(\mu_x,\mu_y)] + E[\frac{\partial f}{\partial X}(\mu_x,\mu_y)(X-\mu_x)] \\ &+ \frac{\partial f}{\partial Y}(\mu_x,\mu_y)(X-\mu_y) \\ &= f(\mu_x,\mu_y) \end{aligned}$$

Now consider a second order Taylor expansion around (x_0, y_0):

$$\begin{aligned} f(x,y) = f(x_0,y_0) &+ \frac{\partial f}{\partial X}(x_0,y_0)(x-x_0) + \frac{\partial f}{\partial Y}(x_0,y_0)(y-y_0) \\ &+ \frac{1}{2}(\frac{\partial^2 f}{\partial X^2}(x_0,y_0)(X-x_0)^2 + \frac{\partial^2 f}{\partial Y^2}(x_0,y_0)(Y-y_0)^2 \\ &+ \frac{\partial^2 f}{\partial X \partial Y}(x_0,y_0)(X-x_0)(Y-y_0)) \end{aligned}$$

We can similar expand $E[f(X,Y)]$ around (μ_x, μ_y) and obtain

$$\begin{aligned} E[f(X,Y)] &= f(\mu_x,\mu_y) + \frac{1}{2}(E[\frac{\partial^2 f}{\partial X^2}(\mu_x,\mu_y)(X-\mu_x)^2] \\ &+ E[\frac{\partial^2 f}{\partial Y^2}(x_0,y_0)(y-y_0))^2] \\ &+ E[\frac{\partial^2 f}{\partial X \partial Y}(\mu_x,\mu_y)(X-\mu_x)(Y-\mu_y)] \\ &= f(\mu_x,\mu_y) + \frac{1}{2}(\frac{\partial^2 f}{\partial X^2}(\mu_x,\mu_y)\mathrm{Var}[X] \\ &+ \frac{\partial^2 f}{\partial Y^2}(x_0,y_0)\mathrm{Var}[Y] \\ &+ \frac{\partial^2 f}{\partial X \partial Y}(\mu_x,\mu_y)\mathrm{Cov}[X,Y] \end{aligned}$$

Plugging in $f(S,C) = S/C$, we have

$$\mathrm{Var}[S/C] \approx (\frac{E[S]^2}{E[C]^2})(\frac{\mathrm{Var}[S]}{E[S]^2} - \frac{2\mathrm{Cov}[S,C]}{E[S]E[C]} + \frac{\mathrm{Var}[C]}{\mathrm{E}[C]^2}) \tag{5.9}$$

Notice that the expression of $E[S]$, $E[C]$, $\mathrm{Var}[S]$ and $\mathrm{Var}[C]$ has already been given in Theorem 5.10, Theorem 5.17. The term $\mathrm{Cov}[X,Y] = E[(X-\mu_x)(Y-\mu_y)]$ can be obtained similar to Theorem 5.10. Hence the theorem follows. □

5.5.3.1 Centralized Sampling for Average

In a centralized setting where a_v, b_v, μ_v and σ_v are given for all v, every term in the expression $\frac{E[S]^2}{E[C]^2}(\frac{\text{Var}[S]}{E[S]^2} - 2\frac{\text{Cov}[S,C]}{E[S]E[C]} + \frac{\text{Var}[C]}{\text{E}[C]^2}$ that depends on p is proportional to either p or $1/p$.[9] The terms proportional to $\frac{1}{p}$ are $\frac{1}{p}(A - 2B + C)$ where

$$\begin{aligned}
A =& \frac{\sum_v a_v(\mu_v^2 + \sigma_v^2)b_v}{(\sum_v a_v\mu_v b_v)^2} + \frac{a_v^2\mu_v^2 b_v^2}{(\sum_v a_v\mu_v b_v)^2} \\
& - \frac{\sum_v a_v^2\mu_v^2 b_v}{(\sum_v a_v\mu_v b_v)^2} - \frac{\sum_v a_v(\mu_v^2 + \sigma_v^2)b_v^2}{(\sum_v a_v\mu_v b_v)^2} \\
B =& \frac{1}{\sum_v a_v b_v} + \frac{\sum_v a_v^2 b_v^2 \mu_v}{(\sum_v a_v b_v)(\sum_v a_v\mu_v b_v)} \\
& - \frac{\sum_v a_v^2\mu_v b_v}{(\sum_v a_v b_v)(\sum_v a_v\mu_v b_v)} - \frac{\sum_v a_v\mu_v b_v^2}{(\sum_v a_v b_v)(\sum_v a_v\mu_v b_v)} \\
C =& \frac{\sum_v a_v b_v}{(\sum_v a_v b_v)^2} + \frac{\sum_v a_v^2 b_v^2}{(\sum_v a_v b_v)^2} - \frac{\sum_v a_v^2 b_v}{(\sum_v a_v b_v)^2} - \frac{\sum_v a_v b_v^2}{(\sum_v a_v b_v)^2}
\end{aligned}$$

The term proportional to p is pD where:

$$D = \frac{1}{\epsilon_2\epsilon_2}(\frac{\sum_v a_v(\mu_v^2 + \sigma_v^2)b_v}{(\sum_v a_v\mu_v b_v)^2} - \frac{2}{\sum_v a_v b_v} + \frac{\sum_v a_v b_v}{(\sum_v a_v b_v)^2}).$$

We can find a p that minimizes $\frac{1}{p}(A-2B+C)+pD$ as follows.

Theorem 5.22. *In the centralized setting, set* $p^- = \max\{\epsilon_1, \epsilon_2\}$, $p^+ = 1$ *and* $p^* = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\frac{A-2B+C}{D}}\}\}$. *Then the optimal sampling parameter is given by:*

$$p = \begin{cases} p^- & \text{if } A - 2B + C \leq 0 \text{ and } D > 0 \\ p^+ & \text{if } A - 2B + C > 0 \text{ and } D \leq 0 \\ p^* & \text{if both } A - 2B + C \text{ and } D > 0 \\ \arg\min_{p\in\{p^-,p^+\}} \frac{1}{p}(A - 2B + C) + pD. & \text{otherwise} \end{cases}$$

Proof. The proof is identical to Theorem 5.11 and Theorem 5.18. □

5.5.3.2 Decentralized Sampling for Average

Minimizing the *worst case* variance for `AVG` (for the decentralized setting) is much more involved than the average case. In most cases, the objective function (variance)

[9] Notice that $E[S]/E[C]$ is independent of p.

is neither convex nor concave in T_2's frequencies. However, note that every term in Theorem 5.22 is an inner product $\langle x, y\rangle$, where x and y are two vectors stored on `party1` and `party2`, respectively. Fortunately, inner products can be approximated by transferring a very small amount of information using the AMS sketch[13, 51]. With such a sketch, we can derive an approximate sampling rate without communicating the full frequency statistics.

5.6 Multiple Queries and Filters

Creating a separate sample for each combination of aggregation function, aggregation column, and `WHERE` clause is clearly impractical. In this section, we show how to create a single sample per join pattern that supports multiple queries at the cost of some possible loss of approximation quality. First, we ignore the `WHERE` clauses and then show how they can be handled too.

Multiple Tables and Queries We formulate our input as a graph $G{=}\langle V, E\rangle$. The vertex set V is the set of all table and join key pairs, and the edge set E corresponds to all join queries of interest. Specifically, for every join query between tables T_1 and T_2 on $J_1 = J_2$, we have a corresponding edge e between vertices $(T_1, J_1) \in V$ and $(T_2, J_2) \in V$ (henceforth, we will use a query and its corresponding edge interchangeably). This means G is a multigraph, with potentially parallel edges or self-loops. For each vertex $v = (T, J) \in V$, we must output a sampling budget ϵ_v as well as the corresponding universe sampling rate p_v, which will be used to create a sample $S = \text{UBS}_{p_v,\epsilon_v/p_v}(T, J)$. This sample will be used for any query that involves a join with T on column(s) J.

According to Lemmas 5.8 and 5.15, and Theorem 5.21, for each edge $e{=}(v_1{,}v_2){\in}E$, we can express the estimator variance of its corresponding query as a function of $\epsilon_{v_1}, \epsilon_{v_2}, p_{v_1}, p_{v_2}$ and p_e, where p_e is an auxiliary variable denoting the minimum of p_1 and p_2:

$$f_e(p, \epsilon_{v_1}, \epsilon_{v_2}, p_{v_1}, p_{v_2}){=}\frac{1}{p_e}(A_e{+}B_e\frac{p_1}{\epsilon_{v_1}}{+}C_e\frac{p_2}{\epsilon_{v_2}}{+}D_e\frac{p_1p_2}{\epsilon_{v_1}\epsilon_{v_2}}) \tag{5.10}$$

where A_e, B_e, C_e, D_e are constants that depend on the distributional information of the tables in v_1 and v_2. To cast this as an optimization problem, we also take in a user specified weight ω_e for each edge e and express our objective as:

$$F = \sum_{e=(v_1,v_2)\in E} \omega_e f_e(p_e, \epsilon_{v_1}, \epsilon_{v_2}, p_{v_1}, p_{v_2}) \tag{5.11}$$

The choice of ω_e values is up to the user. For example, they can all be set to 1, or to the relative frequency, importance, or probability of appearance (e.g., based on past workloads) of the query corresponding to e. Then, to find the optimal sampling parameters we solve the following optimization:

$$\min_{\epsilon_v, p_v, p_e} \quad F \qquad \text{subject to} \sum_{v=(T,J)\in V} \epsilon_v \cdot \text{size}(T) \leq B \tag{5.12}$$

where $size(T)$ is the storage footprint of table T, and B is the overall storage budget for creating samples. Note that by replacing the non-linear $p_e = \min(p_{v_1}, p_{v_1})$ constraints with $p_e \leq p_{v_1}$ and $p_e \leq p_{v_2}$, (5.12) is reduced to a smooth optimization problem, which can be solved numerically with off-the-shelf solvers [21].

Known Filters To incorporate `WHERE` clauses, we simply regard a query with a filter c on $T_1 \bowtie T_2$ as a query without a filter but on a sub-table that satisfies c, namely $T' = \sigma_c(T_1 \bowtie T_2)$.

Unknown Filters with Distributional Information When the columns appearing in the `WHERE` clause can be predicted but the exact constants are unknown, a similar technique can be applied. For example, if an equality constraint $C > x$ is anticipated but x may take on 100 different values, we can *conceptually* treat it as 100 separate queries, each with a different value of x in its `WHERE` clause. This reduces our problem to that of sampling for multiple queries without a `WHERE` clause, which we know how to handle using equation (5.11).[10] Here, the weight ω_i can be used to exploit any distributional information that might be available. In general, ω_i should be set to reflect the probability of each possible `WHERE` clause appearing in the future. For example, if there are R possible `WHERE` clauses and all are equally likely, we can set $\omega_i = 1/R$, but if popular values in a column are more likely to appear in the filters, we can use the column's histogram to assign ω_i.

Unknown Filters When there is no information regarding the columns (or their values) in future filters, we can take a different approach. Since the estimator variance is a monotone function in the frequencies of each join key (see Theorem 5.10, Theorems 5.17 and 5.21), the larger the frequencies, the larger the variance. This means the worst case variance always happens when the `WHERE` clause selects all tuples from the original table. Hence, in the absence of any distributional information regarding

[10]Note that, even though each query in this case is on a different table, they are all sub-tables of the same original table, and hence their sampling rate p is the same.

future `WHERE` clauses, we can simply focus on the original query without any filters to minimize our worst case variance.

5.7 Experiments

Our experiments aim to answer the following questions:

1. How does our optimal sampling compare to other baselines in centralized and decentralized settings? (§5.7.2, §5.7.3)
2. How well does our optimal UBS sampling handle join queries with filters? (§5.7.4)
3. How does our optimal UBS sampling perform when using a single sample for multiple queries? (§5.7.5)
4. How does our optimal SUBS sampling compare to existing stratified sampling strategies? (§5.7.6)
5. How much does a decentralized setting reduce the resource consumption and sample creation overhead? (§5.7.7)
6. How does our optimal sampling compare to a more general sampling scheme, namely two-level sampling [35]? (§5.7.8)

5.7.1 Experiment Setup

Hardware and Software We borrowed a cluster of 18 *c220g5* nodes from CloudLab [3]. Each node was equipped with an Intel Xeon Silver 4114 processor with 10 cores (2.2Ghz each) and 192GB of RAM. We used Impala 2.12.0 as our backend database to store data and execute queries.

Datasets We used several real-life and synthetic datasets:

1. `Instacart` [1]. This is a real-world dataset from an online grocery. We used their *orders* and *order_products* tables (3M and 32M tuples, resp.), joined on *order_id*.
2. `Movielens` [83] This is a real-world movie rating dataset. We used their *ratings* and *movies* tables (27M and 58K tuples, resp.), joined on *movieid*.
3. `TPC-H` [2]. We used a scale factor of 1000, and joined *l_orderkey* of the fact table (*lineitem*, 6B tuples) with *o_orderkey* of the largest dimension table (*orders*, 1.5B tuples).

Table 5.2: Six UBS baselines, each with different p and q.

	B_1	B_2	B_3	B_4	B_5	B_6
p	0.001	0.0015	0.003	0.333	0.6667	1.000
q	1.000	0.6667	0.333	0.003	0.0015	0.001

4. `Synthetic.` To better control the join key distribution, we also generated several synthetic datasets, where tables T_1 and T_2 each had 100M tuples and a join column J. T_1 had an additional column W for aggregation, drawn from a power law distribution with range [1, 1000] and α=3.5. We varied the distribution of the join key in each table to be one of uniform, normal, or power law, creating different datasets (listed in Table 5.3). The values of column J were integers randomly drawn from [1, 10M] according to the chosen distribution. Whenever joining with *power2* (see below), we used 100K join keys in both relations. For normal distribution, we used a truncated distribution with σ=1000/5. We used two different variants of power law distribution for J, one with α=1.5 and 10M join keys (referred to as *power1*), and one with α=2.0 and 100K join keys (referred to as *power2*). We denote each synthetic dataset according to its tables' distributions, $\mathcal{S}\{$*distribution of* T_1*,distribution of* $T_2\}$, e.g., $\mathcal{S}\{$*uniform,uniform*$\}$.

Queries For each dataset, we tested a set of three queries that join two tables and calculate aggregates. The aggregates in each of three queries for a dataset correspond to one of the three most frequently used (non-extreme) aggregates, `COUNT`, `SUM`, and `AVG`, respectively.

Baselines We compared our optimal UBS parameters (referred to as `OPT`) against six baselines. The UBS parameters of these baselines, $B_1, \ldots, B_6$, are listed in Table 5.2. B_1 and B_6 are simply universe and uniform sampling, respectively. $B_2, \ldots, B_5$ represent different hybrid variants of these sampling schemes. Sampling budgets were $\epsilon_1 = \epsilon_2 = 0.001$, except for `Instacart` and `Movielens` where, due to their small number of tuples, we used 0.01 and 0.1, respectively. In Section 5.7.8, we also compare against a more general scheme, called *two-level sampling* [35], which utilizes significantly more parameters than UBS (see Section 5.8 for an overview of two-level sampling).

Implementation We implemented our optimal parameter calculations in Python application. Our sample generation logic read required information, such as table size

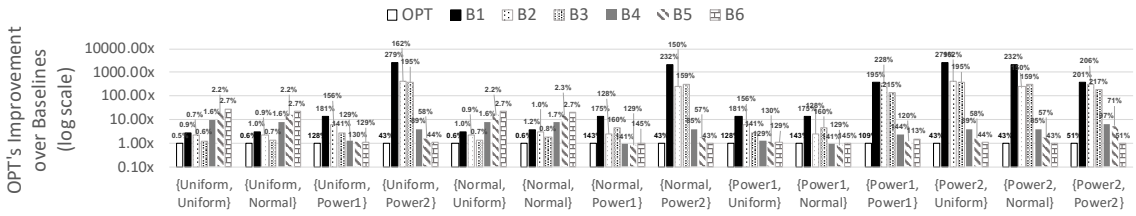


Figure 5.2: `OPT`'s improvement in terms of variance for `COUNT` over six baselines with synthetic dataset (percentages are relative error).

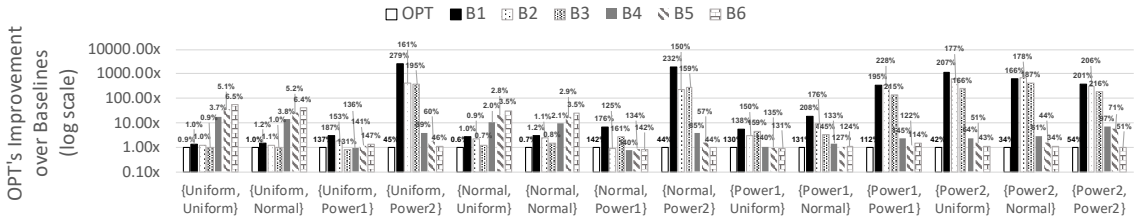


Figure 5.3: `OPT`'s improvement in terms of variance for `SUM` over six baselines with synthetic dataset (percentages are relative error).

and join key frequencies, from the database, and then constructed SQL statements to build appropriate samples in the target database. We used Python to compute approximate answers from sample-based queries.

Variance Calculations We generated β=500 pairs of samples for each experiment, and re-ran the queries on each pair, to calculate the variance of our approximations.

5.7.2 Join Approximation: Centralized Setting

Table 5.3 shows the sampling rates used by `OPT` for each dataset and aggregate function in the centralized setting. For `Synthetic`, the optimal parameters were some mixture of uniform and universe sampling when both tables were only moderately skewed (i.e., uniform or normal distributions) for `COUNT` and `SUM`, whereas it reduced to a simple uniform sampling for power law distribution. This is due to the higher probability of missing extremely popular join keys with universe sampling. To the contrary, for `AVG`, `OPT` reduced to a simple universe sampling in most cases. This is because maximizing the output size in this case was the best way to reduce variance. For the other datasets (`Instacart`, `Movielens`, and `TPC-H`), the optimal parameters led to universe sampling, regardless of aggregate type, and their joins were PK-FK, hence making uniform sampling less useful for the table with primary keys.

Figure 5.2 shows `OPT`'s improvement over the baselines in terms of variance for

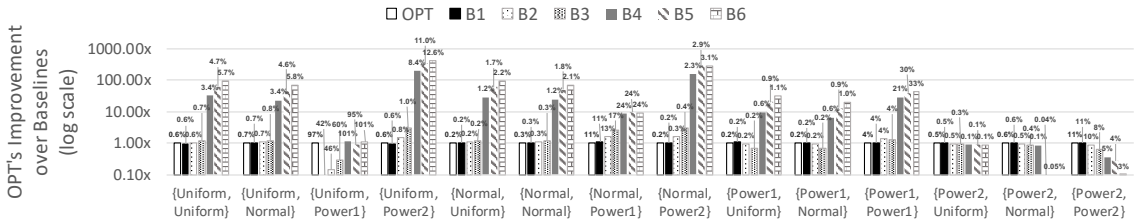
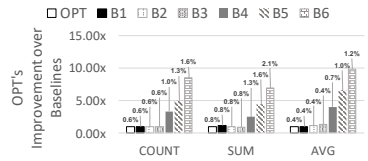
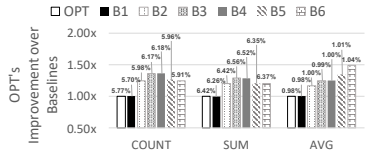


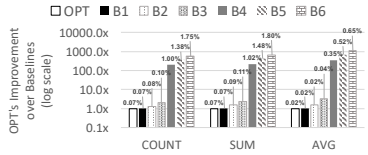
Figure 5.4: `OPT`'s improvement in terms of variance for `AVG` over six baselines with synthetic dataset (percentages are relative error).



(a) `Instacart`



(b) `Movielens`



(c) `TPC-H`

Figure 5.5: `OPT`'s improvement in terms of variance over the baselines on benchmark datasets (percentages are relative error).

`COUNT` queries. Each bar is also annotated with the relative percentage error of the corresponding baseline. `OPT` outperformed all baselines in most cases, achieving over 10x lower variance than the worst baseline. Figures 5.3 and 5.4 show the same experiment for `SUM` and `AVG`. In both cases, `OPT` achieved the minimum variance across all sampling strategies, except for `AVG` when T_1 or T_2 was a power law distribution. This is because `OPT` for `AVG` was calculated using a Taylor approximation, which is accurate only when the estimators of `SUM` and `COUNT` are both within the proximity of their true values. Moreover, sample variance converges slowly to the theoretical variance, particularly for skew distributions, such as power law. This is why estimated variances for `OPT` were not optimal for some `Synthetic` datasets. However, `OPT` still achieved the lowest variance across all real-world datasets, as shown in Figure 5.5. Here, for the selected join key, `OPT` determined that a full universe sampling was the

Table 5.3: Optimal sampling parameters (centralized setting).

Dataset	COUNT		SUM		AVG	
	p	q	p	q	p	q
$\mathcal{S}\{uniform,uniform\}$	0.010	0.1	0.004	0.264	0.001	1.000
$\mathcal{S}\{uniform,normal\}$	0.012	0.083	0.005	0.220	0.001	1.000
$\mathcal{S}\{uniform,power1\}$	1.000	0.001	1.000	0.001	0.692	0.001
$\mathcal{S}\{uniform,power2\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{normal,uniform\}$	0.012	0.083	0.009	0.111	0.001	1.000
$\mathcal{S}\{normal,normal\}$	0.014	0.069	0.011	0.093	0.001	1.000
$\mathcal{S}\{normal,power1\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{normal,power2\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{power1,uniform\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{power1,normal\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{power1,power1\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{power2,uniform\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{power2,normal\}$	1.000	0.001	1.000	0.001	0.001	1.000
$\mathcal{S}\{power2,power2\}$	1.000	0.001	1.000	0.001	0.001	1.000
Instacart	0.01	1.00	0.01	1.00	0.01	1.00
Movielens	0.1	1.00	0.1	1.00	0.1	1.00
TPC-H	0.001	1.00	0.001	1.00	0.001	1.00

best sampling scheme.

In summary, this experiment highlights OPT's ability in outperforming simple uniform or universe sampling—or choosing one of them, when optimal—for aggregates on joins.

5.7.3 Join Approximation: Decentralized

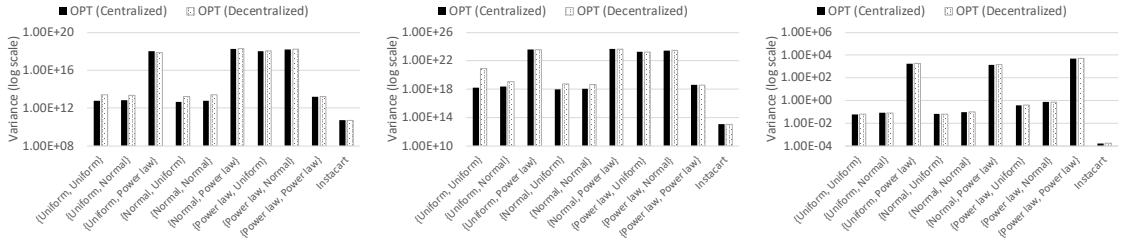
We evaluated both OPT and other baselines under a decentralized setting using Instacart and Synthetic datasets. Here, we constructed a possible worst case distribution for T_2 that was still somewhat realistic, given the distribution of T_1 and minimal information about T_2 (i..e, T_2's cardinality). To do this, we used the following steps: 1) let $J_{MAX(T_1)}$ be the most frequent join key value in T_1; 2) assign 75% of the join key values of T_2 to have the value of $J_{MAX(T_1)}$ and draw the rest of the join key values from a uniform distribution.

Figure 5.6 shows the results. For Synthetic, the OPT was the same under both settings whenever there was a power law distribution or the aggregate was AVG. This is because our assumption of the worst case distribution for T_2 was close to a power law distribution. For COUNT and SUM with Synthetic dataset, OPT in the decentralized setting had a much higher variance than OPT in the centralized setting when there was no power law distribution. With Instacart, OPT in the decentralized setting was

Table 5.4: Optimal sampling parameters for $\mathcal{S}\{uniform,uniform\}$ for different distributions of the filtered column C.

Dist. of C	COUNT		SUM		AVG	
	p	q	p	q	p	q
Uniform	0.010	1.000	0.010	1.000	0.010	1.000
Normal	0.018	0.555	0.015	0.648	0.010	1.000
Power law	0.051	0.195	0.050	0.201	0.010	1.000

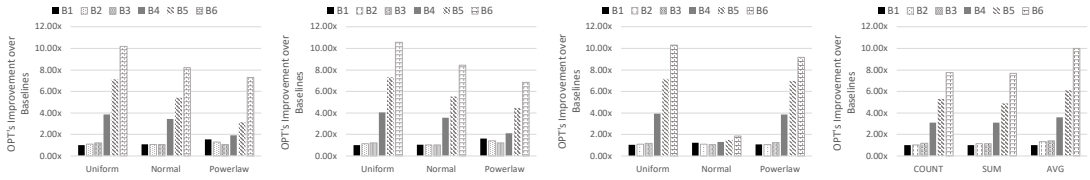
the same as OPT in the centralized setting, which had the minimum variance among the baselines. This illustrates that OPT in the decentralized setting can perform well with real-world data where the joins are mostly PK-FK. This also shows that if a reasonable assumption is possible on the distribution of T_2, OPT can be as effective in the decentralized setting as it is in a centralized one, while requiring significantly less communication.



(a) COUNT (b) SUM (c) AVG

Figure 5.6: Variances of the query estimators for OPT in the centralized and decentralized settings.

5.7.4 Join Approximation with Filters



(a) COUNT (b) SUM (c) AVG (d) Instacart

Figure 5.7: OPT's improvement in terms of the estimator's variance over six baselines in the presence of filters.

To study OPT's effectiveness in the presence of filters, we used $\mathcal{S}\{uniform,uniform\}$ and Instacart datasets with ϵ=0.01. We added an extra column C to T_1 in $\mathcal{S}\{uniform,uniform\}$,

Table 5.5: Sampling parameters (p and q) of OPT using individual samples for different aggregates versus a combined sample ($\mathcal{S}\{normal,normal\}$ dataset).

Scheme	COUNT		SUM		AVG	
	p	q	p	q	p	q
OPT (individual)	0.145	0.069	0.125	0.080	0.010	1.000
OPT (combined)	0.133	0.075	0.133	0.075	0.133	0.075

with integers in $[1, 100]$, and tried three distributions (uniform, normal, power law). For Instacart, we used the *order_hour_of_day* column for filtering, which had an almost normal distribution. We used an equality operator and chose the comparison value x uniformly at random. We calculated the average variance over all possible values of c.

Table 5.4 shows the sampling rates chosen by OPT, while Figure 5.7 shows OPT's improvement over baselines in terms of average variance. Again, OPT successfully achieved the lowest average variance among all baselines in all cases, up to 10x improvement compared to the worst baseline. This experiment confirms that UBS with OPT is highly effective for join approximation, even in the presence of filters.

5.7.5 Combining Samples

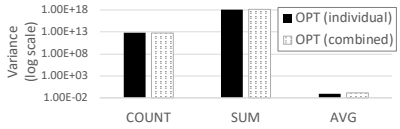
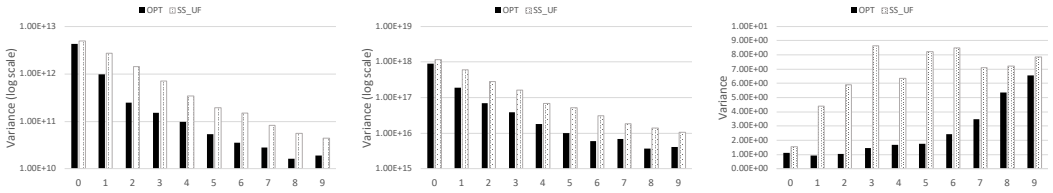


Figure 5.8: Variance of the query estimators for OPT (individual) and OPT (combined) for the $\mathcal{S}\{normal,normal\}$ dataset.

We evaluated the idea of using a single sample for multiple queries instead of generating individual samples for each query, as discussed in Section 5.6. Here, we use OPT (individual) and OPT (combined) to denote the use of one-sample-per-query and one-sample-for-multiple-queries, respectively. For OPT (combined), we considered a scenario where each of COUNT, SUM, and AVG is equally likely to appear. Table 5.5 reports the sampling rates chosen in each case. As shown in Figure 5.8, without having to generate an individual sample for each query, the variances of OPT (combined) were only slightly higher than those of OPT (individual). This experiment shows that it is possible to create a single sample for multiple queries without sacrificing too much optimality.

5.7.6 Stratified Sampling

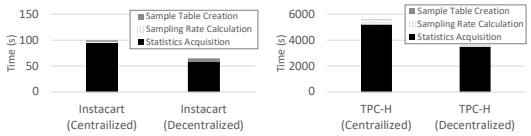


(a) `COUNT` (b) `SUM` (c) `AVG`

Figure 5.9: Query estimator variance per group for for a group-by join aggregate using SUBS versus *SS_UF*.

We also evaluated SUBS for join queries with group-by. Here, we used the $\mathcal{S}\{normal,normal\}$ dataset, and added an extra group column G to T_1 with integers from 0 to 9 drawn from a power law distribution with $\alpha = 1.5$. This time we did not randomize the groups, i.e., G=0 had the most tuples and G=9 had the fewest. This was to study SUBS performance with respect to the different group sizes. As a baseline, we generated stratified samples for T_1 on G with $k_{\text{key}} = 100,000$ and uniform samples for T_2 with a 0.01 sampling budget. We denote this baseline as *SS_UF*. For SUBS, we used parameters that matched the sample size of *SS_UF*, i.e., $k_{\text{key}} = 100, k_{\text{tuple}} = 100,000$. Figure 5.9 shows the variance of query estimators for each of the 10 groups for different aggregations. As expected, SUBS with `OPT` achieved lower variances than *SS_UF* across all aggregates and groups with different sizes.

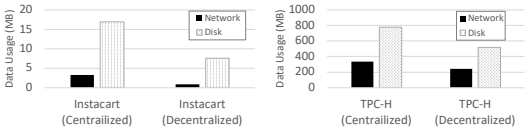
5.7.7 Overhead: Centralized vs. Decentralized



(a) `Instacart` (b) `TPC-H`

Figure 5.10: Time taken to generate samples for `Instacart` and `TPC-H` in centralized vs. decentralized setting.

We compared the overhead of `OPT` in centralized versus decentralized settings, in terms of the sample creation time and resources, such as network and disk. `OPT` should have a much higher overhead in the centralized setting, as it requires full frequency



(a) `Instacart` (b) `TPC-H`

Figure 5.11: Total network and disk bandwidth used to generate samples for `Instacart` and `TPC-H`.

information of every join key value in both tables. To quantify their overhead difference, we used `Instacart` and `TPC-H`, and created a pair of samples for `SUM` in each case. Here, the aggregation type did not matter, as the time spent calculating p and q was negligible compared to the time taken by transmitting the frequency vectors.

As shown in Figure 5.10, we measured the time for statistics acquisition, sampling rate calculation, and sample table creation. Here, the time taken by collecting the frequencies was the dominant factor. For `Instacart`, it took 65.16 secs from start to finish in the decentralized setting, compared to 99.98 secs in the centralized setting, showing 1.53x improvement in time. For `TPC-H`, it took 59.5 min in the decentralized setting, compared to 91.7 mins in the centralized, showing a speedup of 1.54x.

We also measured the total network and disk I/O usage across the entire cluster, as shown in Figure 5.11. For `Instacart`, compared to the decentralized setting, the centralized one used 3.66x (0.9 $\rightarrow$ 3.29 MB) more network and 2.22x (7.59 $\rightarrow$ 16.9 MB) more disk bandwidth. Overall, the overhead was less for `TPC-H`. The centralized in this case used 1.38x (243.39 $\rightarrow$ 337.04 MB) more network and 1.49x (519.03 $\rightarrow$ 776.58 MB) more disk bandwidth than the decentralized setting.

This experiment shows the graceful tradeoff between the optimality of sampling and its overhead, making the decentralized variant an attractive choice for large datasets and distributed systems.

5.7.8 UBS vs. Two-Level Sampling

Two-level sampling (`2LV`) [35] is similar to our UBS scheme in that it also applies stratified sampling before Bernoulli sampling. However, unlike UBS which applies the same sampling rate to all tuples, `2LV` uses a different universe sampling rate for each join key, i.e., `2LV` is strictly more expressive than UBS. Thus, by using significantly more parameters (i.e., number of distinct join keys), `2LV` should be able to achieve a lower variance than any UBS scheme (which uses only two parameters p and q). To empirically measure this gap, we compared the relative error of `2LV` versus `OPT`. Since

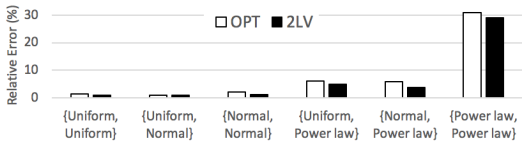


Figure 5.12: Optimal UBS vs. two-level sampling

`2LV` is originally designed for cardinality estimation, we compared their `COUNT` error on our `Synthetic` datasets (since `COUNT` is symmetric, we only have 6 combinations), with 100K tuples, 100 keys, and ϵ=1% budget. Here, we used As shown in Figure 5.12, `2LV`'s error was slightly lower than `OPT`, and both errors increased with the skew in the join keys, e.g., when a power law was involved. This was as expected: having a separate parameter for each join key means more complexity, but also allows `2LV` to better adopt to the distribution of the data.

5.8 Related Work

Online Sample-based Join Approximation Ripple Join [80] is an online join algorithm that operates under the assumption that the tuples of the original tables are processed in a random order. Each time, it retrieves a random tuple (or a set of random tuples) from the tables, and then joins the new tuples with the previously read tuples and with each other. SMS [90] speeds up the hashed version of Ripple Join when hash tables exceed memory. Wander Join [107] tackles the problem of k-way chain join and eliminates the random order requirement of Ripple Join. However, it requires an index on every join column in each of the tables. Using indexes, Wander Join performs a set of random walks and obtains a *non-uniform* but independent sample of the join. Maintaining an approximation of the size of all partial joins can help overcome the non-uniformity problem [107].

Offline Sample-based Join Approximation AQUA [6] acknowledges the quadratic reduction and the non-uniformity of the output when joining two uniform random samples. The same authors propose *Join Synopsis* [8], which computes a sample of one of the tables and joins it with the other tables as a sample of the actual join. Chaudhuri et al. [30] also point out that a join of independent samples from two relations does not yield an independent sample of their join, and propose using pre-computed statistics to overcome this problem. However, their solution can be quite costly, as it requires collecting full frequency information of the relation. Zhao et al. [107] provide a better trade-off between sampling efficiency and the join size upper

bound. Hashed sampling (a.k.a. universe) [81] is proposed in the context of selectivity estimation for set similarity queries. Block-level uniform sampling [31] is less accurate but more efficient than tuple-level sampling. Bi-level sampling [36] performs Bernoulli sampling at both the block- and tuple-level, as a trade-off between accuracy and I/O cost of sample generation. Kamat and Nandi [92] use simple stratified sampling on join column but with an objective function measuring the amount of randomness of the sample scheme, which shows improvement over simple correlated sampling.

AQP Systems on Join Most AQP systems rely on sampling and support certain types of joins [93, 119, 6, 32, 11, 77, 107]. STRAT [32] discusses the use of uniform and stratified sampling, and how those can support certain types of join queries. More specifically, STRAT only supports PK-FK joins between a fact table and one or more dimension table(s). BlinkDB [11] extends STRAT and considers multiple stratified samples instead of a single one. As previously mentioned, AQUA [6] supports foreign key joins using join synopses. *Icicles* [77] samples tuples that are more likely to be required by future queries, but, similar to AQUA, only supports foreign key joins. PF-OLA [124] is a framework for parallel online aggregation. It studies parallel joins with group-bys, when partitions of the two tables fit in memory. XDB [107] integrates Wander Join in PostgreSQL. Quickr [93] does not create offline samples. Instead, it uses universe sampling to support equi-joins, where the group-by columns and the value of aggregates are not correlated with the join keys. VerdictDB [119] is a universal AQP framework that supports all three types of samples (uniform, universe, and stratified). VerdictDB utilizes a technique called *variational subsampling*, which creates subsamples of the sample such that it only requires a single join—instead of repeatedly joining the subsamples multiple times—to produce accurate aggregate approximations. ApproxJoin [126] uses Bloom Filters in conjunction with stratified sampling to efficiently produce a sample to the join when relations are distributed across different nodes.

Join Cardinality Estimation There is extensive work on join cardinality estimation (i.e.,`count(*)`) in the database community [139, 141, 62, 14, 133, 106, 100] as an important step of the query optimization process for joins. Two-level sampling [35] first applies universe sampling to the join values, and then, for each join value sampled, it performs Bernoulli sampling. However, unlike our UBS scheme which applies the same rate to all keys, two-level sampling uses a different rate dur-

ing its universe sampling for each join key. In other words, two-level sampling is a more complex scheme with significantly more parameters than UBS (which requires only two parameters, p and q), and is thus less amenable to efficient and decentralized implementation. Furthermore, two-level sampling applies two different sampling methods, whereas bi-level sampling [36] uses only Bernoulli sampling but at different granularity levels. End-biased sampling [62] samples each tuple with a probability proportional to the frequency of its join key. Index-based sampling [106] and deep learning [100] have also been utilized to improve cardinality estimates.

Theoretical Studies The question about the limitation of sample-based approximation of joins, to the best of our knowledge, has not been asked in the theory community. However, the past work in communication complexity on set intersection and inner product estimation has implications for join approximation. In this problem, the Alice and Bob possess respectively two vectors x and y and they wish to compute their inner product $t = \langle x, y \rangle$ without exchanging the vector x and y. In the one-way model, Alice computes a summary $\beta(x)$ and sends it to Bob, who will estimate $\langle x, y \rangle$ using y and $\beta(x)$. For this problem, [117] shows that any estimator produced by s bits of communication has variance at least $\Omega(dt/s)$. Estimating inner product for $0, 1$ vectors is directly related to estimating `SUM` and `COUNT` for a PK-FK join. A natural question is whether the join is still hard even if frequencies are all larger than 1. Further, the question of whether estimating `AVG` is also hard is not answered by prior work.

BIBLIOGRAPHY

[1] The instacart online grocery shopping dataset 2017. https://www.instacart.com/datasets/grocery-shopping-2017. Accessed: 2019-07-20.

[2] TPC-H Benchmark. http://www.tpc.org/tpch/.

[3] Cloudlab. https://www.cloudlab.us, 2019.

[4] A. Abboud, K. Censor-Hillel, S. Khoury, and C. Lenzen. Fooling views: A new lower bound technique for distributed computations under congestion. *CoRR*, abs/1711.01623, 2017.

[5] I. Abraham, Y. Bartal, and O. Neiman. Local embeddings of metric spaces. *Algorithmica*, 72(2):539–606, 2015.

[6] S. Acharya, P. B. Gibbons, and V. Poosala. Aqua: A fast decision support systems using approximate query answers. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB)*, pages 754–757, 1999.

[7] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *Proceedings of the 1999 ACM International Conference on Management of Data, SIGMOD*, pages 574–576, 1999.

[8] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *Proceedings of the 1999 ACM International Conference on Management of Data, SIGMOD*, pages 275–286, 1999.

[9] P. K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In D. B. Shmoys, editor, *Proceeding of Symposium on Theory of Computing, STOC*, pages 555–564. ACM, 2014.

[10] S. Agarwal, A. Panda, B. Mozafari, A. P. Iyer, S. Madden, and I. Stoica. Blink and it's done: Interactive queries on very large data. *PVLDB*, 5(12):1902–1905, 2012.

[11] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Eighth Eurosys Conference 2013, EuroSys '13*, pages 29–42, 2013.

[12] T. Akiba, C. Sommer, and K. Kawarabayashi. Shortest-path queries for complex networks: exploiting low tree-width outside the core. In *Proceedings of 15th International Conference on Extending Database Technology, EDBT*, pages 144–155, 2012.

[13] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[14] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002.

[15] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[16] S. Arora. Nearly linear time approximation schemes for euclidean TSP and other geometric problems. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 554–563, 1997.

[17] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. *SIAM J. Comput.*, 42(4):1737–1767, 2013.

[18] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM International Conference on Management of Data, SIGMOD*, pages 539–550, 2003.

[19] L. Barenboim and M. Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing*, 22 (5-6):363–379, 2010.

[20] Y. Bartal, L. Gottlieb, and R. Krauthgamer. The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme. *SIAM J. Comput.*, 45(4):1563–1581, 2016.

[21] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014.

[22] J. Brody, A. Chakrabarti, R. Kondapally, D. P. Woodruff, and G. Yaroslavtsev. Beyond set disjointness: the communication complexity of finding the intersection. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 106–113, 2014.

[23] J. Brody, A. Chakrabarti, R. Kondapally, D. P. Woodruff, and G. Yaroslavtsev. Certifying equality with limited interaction. *Algorithmica*, 76(3):796–845, 2016.

[24] H. Buhrman, D. García-Soriano, A. Matsliah, and R. de Wolf. The non-adaptive query complexity of testing k-parities. *Chicago J. Theor. Comput. Sci.*, 2013, 2013.

[25] T. H. Chan and A. Gupta. Approximating TSP on metrics with bounded global growth. *SIAM J. Comput.*, 41(3):587–617, 2012.

[26] Y. Chang and T. Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 66–73, 2019.

[27] Y. Chang, S. Pettie, and H. Zhang. Distributed triangle detection via expander decomposition. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840, 2019.

[28] K. Chatterjee and J. Lacki. Faster algorithms for markov decision processes with low treewidth. In *Proceedings of Computer Aided Verification - 25th International Conference, CAV*, pages 543–558, 2013.

[29] A. Chattopadhyay and T. Pitassi. The story of set disjointness. *SIGACT News*, 41(3):59–85, 2010.

[30] S. Chaudhuri, R. Motwani, and V. R. Narasayya. On random sampling over joins. In *Proceedings of the 19999 ACM International Conference on Management of Data, SIGMOD*, pages 263–274, 1999.

[31] S. Chaudhuri, G. Das, and U. Srivastava. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM International Conference on Management of Data, SIGMOD*, pages 287–298, 2004.

[32] S. Chaudhuri, G. Das, and V. R. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2):9, 2007.

[33] C. Chekuri and K. Quanrud. Approximating the Held-Karp bound for metric TSP in nearly-linear time. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 789–800, 2017.

[34] C. Chekuri and K. Quanrud. Fast approximations for metric-tsp via linear programming. *CoRR*, abs/1802.01242, 2018. URL http://arxiv.org/abs/1802.01242.

[35] Y. Chen and K. Yi. Two-level sampling for join size estimation. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 759–774, 2017.

[36] Y. Cheng, W. Zhao, and F. Rusu. Bi-level online aggregation on raw data. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 10:1–10:12, 2017.

[37] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985.

[38] K. Choromanski, T. Jebara, and K. Tang. Adaptive anonymity via b-matching. In *Advances in Neural Information Processing Systems 27: 27th Annual Conference on Neural Information Processing System, NIPS 2013*, pages 3192–3200, 2013.

[39] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[40] R. Cole, R. Hariharan, M. Lewenstein, and E. Porat. A faster implementation of the goemans-williamson clustering algorithm. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[41] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. Mapreduce online. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, pages 313–328, 2010.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[43] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.

[44] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[45] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. Vizdom: Interactive analytics through pen and touch. *PVLDB*, 8(12):2024–2027, 2015.

[46] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[47] M. Cygan, H. N. Gabow, and P. Sankowski. Algorithmic applications of baur-strassen's theorem: Shortest cycles, diameter, and matchings. *J. ACM*, 62(4): 28:1–28:30, 2015.

[48] A. Czumaj and C. Konrad. Detecting cliques in CONGEST networks. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, volume 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:15, 2018.

[49] A. Dasgupta, R. Kumar, and D. Sivakumar. Sparse and lopsided set disjointness via information theory. In *Proceedings of the 15th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX)*, pages 517–528, 2012.

[50] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *Proceedings of 10th Annual European Symposium (ESA)*, pages 323–334, 2002.

[51] A. Dobra, M. N. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 61–72, 2002.

[52] A. Dobra, C. Jermaine, F. Rusu, and F. Xu. Turbo-charging estimate convergence in DBO. *PVLDB*, 2(1):419–430, 2009.

[53] D. E. Drake and S. Hougardy. A simple approximation algorithm for the weighted matching problem. *Inf. Process. Lett.*, 85:211–213, 2003.

[54] A. Drucker, F. Kuhn, and R. Oshman. On the power of the congested clique model. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 367–376, 2014.

[55] R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *J. ACM*, 61:1:1–1:23, 2014.

[56] R. Duan, S. Pettie, and H. Su. Scaling algorithms for weighted matching in general graphs. *ACM Trans. Algorithms*, 14:8:1–8:35, 2018.

[57] R. Duan, S. Pettie, and H.-H. Su. Scaling algorithms for weighted matching in general graphs. *ACM Transactions on Algorithms*, 14, 2018.

[58] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[59] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17: 449–467, 1965.

[60] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards: Section B Mathematics and Mathematical Physics*, 69B:125–130, 1965.

[61] J. Edmonds and E. L. Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop*, pages 27–30, 2001.

[62] C. Estan and J. F. Naughton. End-biased samples for join cardinality estimation. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE*, page 20, 2006.

[63] T. Feder, E. Kushilevitz, M. Naor, and N. Nisan. Amortized communication complexity. *SIAM J. Comput.*, 24(4):736–750, 1995.

[64] O. Fischer, T. Gonen, F. Kuhn, and R. Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 153–162, 2018.

[65] F. V. Fomin, D. Lokshtanov, S. Saurabh, M. Pilipczuk, and M. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Trans. Algorithms*, 14(3):34:1–34:45, 2018.

[66] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. ACM*, 31(3):538–544, 1984.

[67] H. Gabow and S. Pettie. The dynamic vertex minimum problem and its application to clustering-type approximation algorithms. In *Proceedings 8th Scandinavian Workshop on Algorithm Theory (SWAT), LNCS Vol. 2368*, pages 190–199, 2002.

[68] H. N. Gabow. An efficient implementation of edmonds' algorithm for maximum matching on graphs. *J. ACM*, 23(2):221–234, 1976.

[69] H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC*, pages 448–456, 1983.

[70] H. N. Gabow. Scaling algorithms for network problems. *J. Comput. Syst. Sci.*, 31, Sept. 1985.

[71] H. N. Gabow. Data structures for weighted matching and extensions to b-matching and f-factors. *ACM Trans. Algorithms*, 14:39:1–39:80, 2018.

[72] H. N. Gabow and P. Sankowski. Algebraic algorithms for b-matching, shortest undirected paths, and f-factors. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 137–146, 2013.

[73] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *J. Comput. Syst. Sci.*, 30:209–221, 1985.

[74] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18:1013–1036, 1989.

[75] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM*, 38:815–853, 1991.

[76] Z. Galil, S. Micali, and H. N. Gabow. Priority queues with variable priority and an O(EV log V) algorithm for finding a maximal weighted matching in general graphs. In *Proceedings of the 23th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 255–261, 1982.

[77] V. Ganti, M. Lee, and R. Ramakrishnan. ICICLES: self-tuning samples for approximate query answering. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases*, pages 176–187, 2000.

[78] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[79] T. Gonen and R. Oshman. Lower bounds for subgraph detection in the CONGEST model. In *Proceedings of the 21st International Conference on Principles of Distributed Systems (OPODIS)*, volume 95 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:16, 2018.

[80] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation. In *Proceedings of the 1999 ACM International Conference on Management of Data, SIGMOD*, pages 287–298, 1999.

[81] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava. Hashed samples: selectivity estimators for set similarity selection queries. *PVLDB*, 1(1):201–212, 2008.

[82] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57(3):366–375, 1998.

[83] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *TiiS*, 5(4):19:1–19:19, 2016.

[84] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1): 105–142, 1999.

[85] J. Håstad and A. Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[86] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proceedings of the 1997 ACM International Conference on Management of Data, SIGMOD*, pages 171–182, 1997.

[87] K. Hildrum, J. Kubiatowicz, S. Ma, and S. Rao. A note on the nearest neighbor in growth-restricted metrics. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2004*, pages 560–561, 2004.

[88] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47, 1952.

[89] T. Izumi and F. L. Gall. Triangle finding and listing in CONGEST networks. In *Proceedings of the 36th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 381–389, 2017.

[90] C. Jermaine, A. Dobra, S. Arumugam, S. Joshi, and A. Pol. A disk-based join with probabilistic guarantees. In *Proceedings of the 2005 ACM International Conference on Management of Data, SIGMOD*, pages 563–574, 2005.

[91] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

[92] N. Kamat and A. Nandi. A unified correlation-based approach to sampling over joins. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 20:1–20:12, 2017.

[93] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. Quickr: Lazily approximating complex adhoc queries in bigdata clusters. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD*, pages 631–646, 2016.

[94] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 741–750, 2002.

[95] K. Kawarabayashi and B. A. Reed. A nearly linear time algorithm for the half integral disjoint paths packing. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 446–454, 2008.

[96] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 217–226, 2014.

[97] A. Khan, K. Choromanski, A. Pothen, S. M. Ferdous, M. Halappanavar, and A. Tumeo. Adaptive anonymization of data using b-edge cover. In *Proceedings of the International Conference for High Performance Computing,Networking, Storage, and Analysis, SC*, pages 59:1–59:11, 2018.

[98] A. M. Khan and A. Pothen. A new 3/2-approximation algorithm for the b-edge cover problem. In *2016 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing, CSC*, pages 52–61, 2016.

[99] D. Kim, L. Liu, S. I. In-Su, J. Kim, and K. Han. Spatial tinydb: A spatial sensor database system for the USN environment. *IJDSN*, 9, 2013.

[100] A. Kipf, T. Kipf, B. Radke, V. Leis, P. A. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. In *Proceedings of the 9th Biennial Conference on Innovative Data Systems Research, CIDR*, 2019.

[101] J. H. Korhonen and J. Rybicki. Deterministic subgraph detection in broadcast CONGEST. In *Proceedings of the 21st International Conference on Principles of Distributed Systems (OPODIS)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 4:1–4:16, 2018.

[102] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In *Distributed Computing*, pages 273–287, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[103] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[104] R. Kyng, R. Peng, S. Sachdeva, and D. Wang. Flows in almost linear time via adaptive preconditioning. In *Proceedings of the 51st Annual Symposium on Theory of Computing, STOC*, pages 902–913, 2019.

[105] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Books on Mathematics Series. 2001. ISBN 9780486414539.

[106] V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann. Cardinality estimation done right: Index-based join sampling. In *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research, CIDR*, 2017.

[107] F. Li, B. Wu, K. Yi, and Z. Zhao. Wander join and XDB: online aggregation via random walks. *ACM Trans. Database Syst.*, 44(1):2:1–2:41, 2019.

[108] L. Lovasz. Communication complexity: A survey. Technical Report TR-204-89, Computer Science Dept., Princeton University, 1989.

[109] S. Maniu, P. Senellart, and S. Jog. An experimental study of the treewidth of real-world graph data. In *22nd International Conference on Database Theory, ICDT*, pages 12:1–12:18, 2019.

[110] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[111] B. Mozafari. Approximate query engines: Commercial challenges and research opportunities. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD*, pages 521–524, 2017.

[112] B. Mozafari and N. Niu. A handbook for building an approximate query engine. *IEEE Data Eng. Bull.*, 2015.

[113] B. Mozafari, E. Z. Y. Goh, and D. Y. Yoon. Cliffguard: A principled framework for finding robust database designs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1167–1182, 2015.

[114] V. Nikishkin. Amortized communication complexity of an equality predicate. In *Computer Science – Theory and Applications*, pages 212–223, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[115] N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices. *J. Economic Theory*, 129(1):192–224, 2006.

[116] L. Orecchia and N. K. Vishnoi. Towards an sdp-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 532–545, 2011.

[117] R. Pagh, M. Stöckel, and D. P. Woodruff. Is min-wise hashing optimal for summarizing set intersection? In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 109–120, 2014.

[118] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie. Online aggregation for large mapreduce jobs. *PVLDB*, 4(11):1135–1145, 2011.

[119] Y. Park, B. Mozafari, J. Sorenson, and J. Wang. Verdictdb: Universalizing approximate query processing. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD*, pages 1461–1476, 2018.

[120] S. Pettie and P. Sanders. A simpler linear time $2/3 - \epsilon$ approximation for maximum weight matching. *Inf. Process. Lett.*, 91:271–276, 2004.

[121] L. Planken, M. M. de Weerdt, and R. van der Krogt. Computing all-pairs shortest paths by leveraging low treewidth. *CoRR*, abs/1401.4609, 2014.

[122] R. Preis. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *Proceedings of 16th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, pages 259–269, 1999.

[123] W. Pulleyblank. *Faces of matching polyhedra*. PhD thesis, University of Waterloo, Ontario, Canada, 1973.

[124] C. Qin and F. Rusu. PF-OLA: a high-performance framework for parallel online aggregation. *Distributed and Parallel Databases*, 32(3):337–375, 2014.

[125] K. Quanrud. Nearly linear time approximations for mixed packing and covering problems without data structures or randomization. In *3rd Symposium on Simplicity in Algorithms, SOSA@SODA*, pages 69–80, 2020.

[126] D. L. Quoc, I. E. Akkus, P. Bhatotia, S. Blanas, R. Chen, C. Fetzer, and T. Strufe. Approxjoin: Approximate distributed joins. In *Proceedings of the ACM Symposium on Cloud Computing, SoCC*, pages 426–438, 2018.

[127] A. Rao and A. Yehudayoff. Communication complexity. (unpublished manuscript; available from the authors' homepages), 2019.

[128] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[129] T. Roughgarden. Communication complexity (for algorithm designers). *Foundations and Trends in Theoretical Computer Science*, 11(3-4):217–404, 2016.

[130] M. Sağlam and G. Tardos. On the communication complexity of sparse set disjointness and exists-equal problems. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 678–687, 2013.

[131] J. P. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM J. Comput.*, 19(5):775–786, 1990.

[132] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics. Springer, 2002. ISBN 9783540443896.

[133] A. N. Swami and K. B. Schiefer. On the estimation of join result sizes. In *Proceedings of 4th International Conference on Extending Database Technology EDBT*, pages 287–300, 1994.

[134] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22:215–225, 1975.

[135] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal of the London Mathematical Society*, s1-36:221–230, 1961.

[136] K. R. Varadarajan and P. K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 805–814, 1999.

[137] V. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ general graph maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.

[138] V. V. Vazirani. An improved definition of blossoms and a simpler proof of the MV matching algorithm. *CoRR*, abs/1210.4594, 2012.

[139] D. Vengerov, A. C. Menck, M. Zaït, and S. Chakkappen. Join size estimation subject to filter conditions. *PVLDB*, 8(12):1530–1541, 2015.

[140] S. Wu, B. C. Ooi, and K. Tan. Continuous sampling for online aggregation over multiple queries. In *Proceedings of the 2010 ACM International Conference on Management of Data, SIGMOD*, pages 651–662, 2010.

[141] W. Wu, J. F. Naughton, and H. Singh. Sampling-based query re-optimization. In *Proceedings of the 2016 ACM International Conference on Management of Data, SIGMOD*, pages 1721–1736, 2016.

[142] A. C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[143] A. C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

[144] D. Y. Yoon, M. Chowdhury, and B. Mozafari. Distributed lock management with RDMA: decentralization without starvation. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD*, pages 1571–1586, 2018.

[145] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.